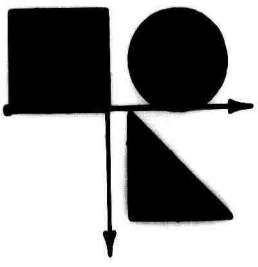
Massachusetts

COMPUTER ASSOCIATES

division of

APPLIED DATA RESEARCH, INC.



APPLIED DATA RESEARCH, INC.

LAKESIDE OFFICE PARK, WAKEFIELD, MASSACHUSETTS 01880 · (617) 245-9540

FINAL REPORT - TASK AREA I
(Volume III)

(21 June 1968 - 31 December 1970)

FOR THE PROJECT
RESEARCH IN MACHINE-INDEPENDENT
SOFTWARE PROGRAMMING

Principal Investigators:

Task Area I	Carlos Christensen	(617) 245-9540
Task Area II	Anatol W. Holt	(617) 245-9540

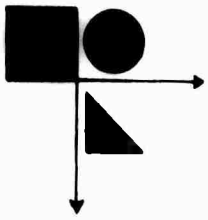
Project Manager:
Robert E. Millstein (617) 245-9540

ARPA Order Number - ARPA 1228
Program Code Number - 8D30

Contractor:	Massachusetts Computer Associates, Inc., Division of ADR
Contract No.:	DAHC04-68-C-0043
Effective Date:	21 June 1968
Expiration Date:	30 September 1971
Amount:	$696,800.00

Sponsored by
Advanced Research Projects Agency
ARPA Order Number - 1228

CA-7102-2613



APPLIED DATA RESEARCH, INC.

LAKESIDE OFFICE PARK, WAKEFIELD, MASSACHUSETTS 01880 · (617) 245-9540

A REPORT ON AMBIT/G
(Volume III)

by

Carlos Christensen

Michael S. Wolfberg

Michael J. Fischer*

CA-7102-2613
February 26, 1971

*Consultant to Applied Data Research, Inc.
Address: Department of Mathematics, M.I.T.,
Cambridge, Massachusetts

This is the third of four volumes of the final report on Task Area I of the project "Research in Machine-Independent Software Programming". This research was supported by the Advanced Research Projects Agency of The Department of Defense and was monitored by U.S. Army Research Office-Durham, Box CM, Duke Station, Durham, North Carolina 27706, under Contract DAHC04-68-C-0043.

CONTENTS

CHAPTER 12

THE AMBIT/G INTERPRETER AS AN AMBIT/G PROGRAM

This volume of "A Report on AMBIT/G" describes the AMBIT/G interpreter as an AMBIT/G program, which is, in fact, how it is implemented. We begin by giving a short verbal description of the interpreter where each function of the interpreter is briefly discussed. The remainder of the volume consists of many pages of diagrams which constitute the listing of the interpreter.

The reader who wants to understand the program (rather than just look at the pretty pictures) is urged to be familiar with the contents of Chapters 3 and 4 of this report in Volume I. Although we have attempted to describe completely what the interpreter does, it is the listing of the interpreter which can be used as a final reference for obscure issues. Furthermore, this listing contributes to a formal definition of AMBIT/G. The listing begins with definitions of characteristic shapes of built-in nodes used by the interpreter and links used by the interpreter.

A table of contents for the listing is given at the beginning of the listing.

A DESCRIPTION

The AMBIT/G interpreter consists of a main program and ten major functions. In addition, the interpreter includes calls on functions 'tail_1', 'tail_2' and 'head_1' which are used to transmit arguments and results. The listing of the interpreter does not include the definitions of these functions. Another kind of omission in the listing is the write calls on built-ins 'read_function' and 'write_function' to define the functions of the interpreter.

Since Chapter 4 (in Volume I) described the interesting actions of the interpreter, we shall only include here individual descriptions of its ten major functions.

getnode

This function is used to get a node off of a free list. Its first argument is the head node of some free list, and its second argument is the terminator node for the list. The result of the function is a node off of the front of the list if there is one. Otherwise, 'locate' is used to find a node of the appropriate type which is the result of the function.

freelist

This function is called as a write function with two tail arguments and a head argument. Its first tail argument is the head node of some free list, and its second tail argument is the terminator node for the list. The head argument is a list of zero or more nodes of the type corresponding to the tail arguments; the effect of this function is to return this list to the free list specified by the tail arguments.

arglist

This function is used to prepare a 'tails' argument list or a 'heads' argument list for making a call on a user function. Its first argument is a 'diamond' which is the destination of either the 'org' link or 'dest' link of the 'linkrep' which represents the user function call about to be made. The second argument of 'arglist' is either 'flag def' or 'flag undef' . If the second argument is 'flag def', this function forms as its result, a list of 'pipe' nodes. The 'value' link of each 'pipe' node points to a user node which is the destination of the 'rep' link of the 'noderep' pointed to by the corresponding 'diamond' of the list supplied as first argument. If, however, the second argument is 'flag undef', this function forms as its result a list of 'pipe' nodes whose 'value' links point to the node 'undef undef'. The result list then has the same length as the length of the list supplied as first argument.

typelist

This function is used to prepare a list of 'type' nodes connected by 'cell's for a read call on either the built-in 'read_function' or 'write_function'.

Its single argument is a 'diamond' which is the destination of the 'org' link of the 'linkrep' which represents the link being processed. Its result is a list of 'cell's, where each 'cell' corresponds to a 'diamond' in the(possibly null) list given as argument. The 'value' link of each 'cell' points to the 'type' node corresponding to the type of the destination of the 'rep' link of the 'noderep' pointed to by the corresponding 'diamond'.

vset

This function is called as a write function to verify or to set (bind) a 'rep' link of a 'noderep' pointed to by a 'diamond' in a list of 'diamond's hanging off the 'dest' link of a 'linkrep' node. Its first tail argument is the 'noderep' in question, and its second tail argument is the 'diamond' in question which points to that 'noderep'. The one head argument is the user node to be considered. If the 'sets' link of the 'noderep' points to the 'diamond', the 'rep' link of the 'noderep' is set to point at the user node. Otherwise, the destination of the 'rep' link is checked to see if it is the user node; if not, 'circle proceed' is made to point at 'flag no'.

lvset

This function is called as a write function to apply the 'vset' function individually to several sets of arguments which correspond to all heads of a read call made on a user function. Its one tail argument is the 'diamond' which is the destination of the 'dest' link of the 'linkrep' representing the user function call which just occurred. Its one head argument is the list of 'pipe's used to transmit the (possibly zero) results of the user function. If the two lists do not have the same number of elements, an error condition occurs. The function ends by returning the given 'pipe' list to the 'pipe' free list.

compile

This function is called with one argument which is the 'rule' node which is to be compiled. A section in the chapter on the interpreter (in Vol. I) described this function in detail.

initnode

This function is called by the compiler to initialize the 'noderep' which is its only argument. If the 'variability' link of the 'noderep' points to 'flag fixed', then its 'sets' link is made to point to 'diamond matched'; or if the 'variability' link of the 'noderep' points to 'flag dummy', then its 'sets' link is made to point to 'diamond unmatched'. Otherwise, an error condition occurs.

initlist

This function is called by the compiler to initialize a list of 'noderep's connected by 'diamond's. The 'diamond' which heads such a (possibly empty) list is the only argument. The 'initnode' function is applied to every 'noderep' in the list.

allmatch

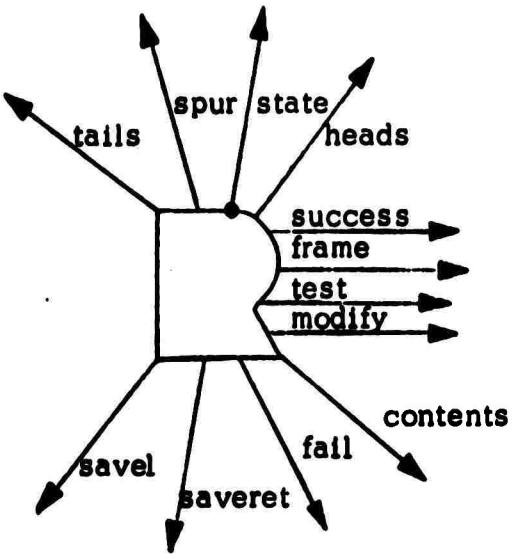
This function is called by the compiler to check whether all 'nodrep's have been matched on a list of 'noderep's connected by 'diamond's. Its one argument is a 'diamond' which heads such a list. If any 'noderep' in the (possibly empty) list has a 'sets' link pointing to 'diamond unmatched' then the one result is 'diamond unmatched'; otherwise, it is 'diamond matched'.

THE LISTING

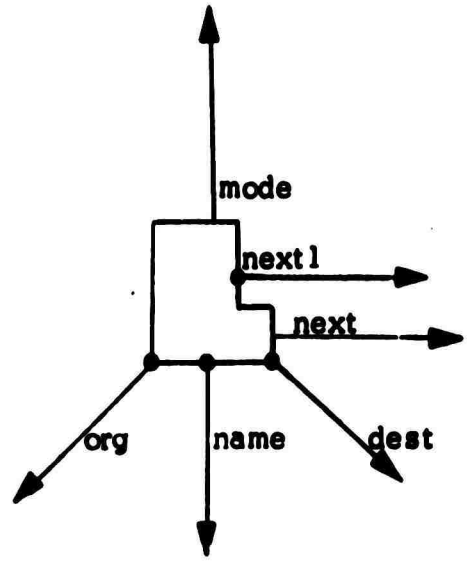
The remainder of this volume consists of the listing of the AMBIT/G interpreter. The following is a table of contents of the listing.

* Note that the loader page titles appear in the upper-right corner of each page.

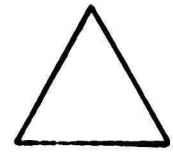
rule :


tails
spur
state
heads
success
frame
test
modify
contents
fail
savel
saveret

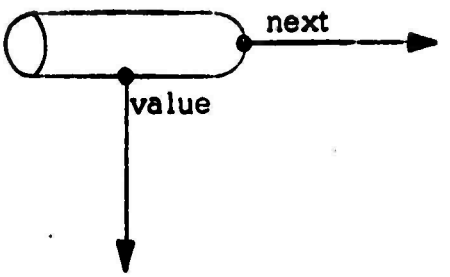
linkrep :



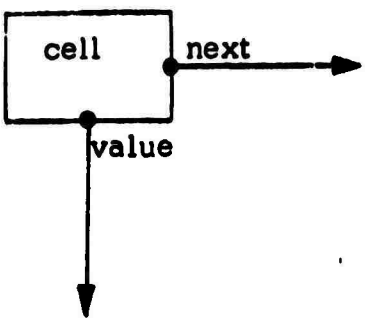
flag:



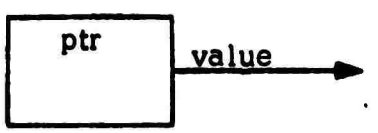
pipe :



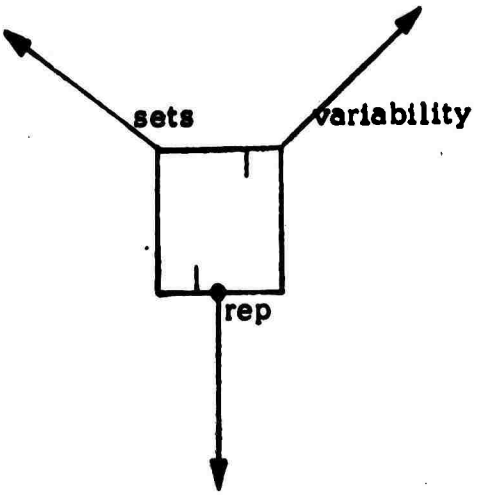
cell :



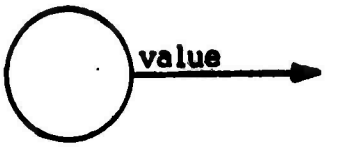
ptr :



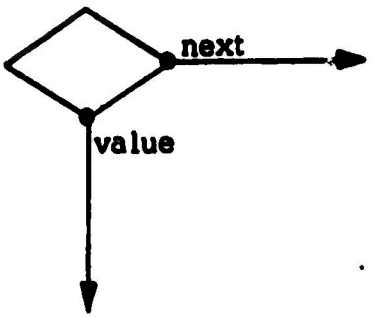
noderep :



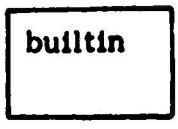
circle :



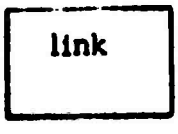
diamond :



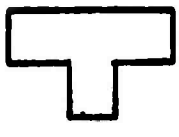
builtin :



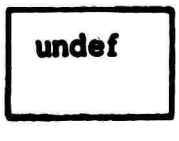
link:

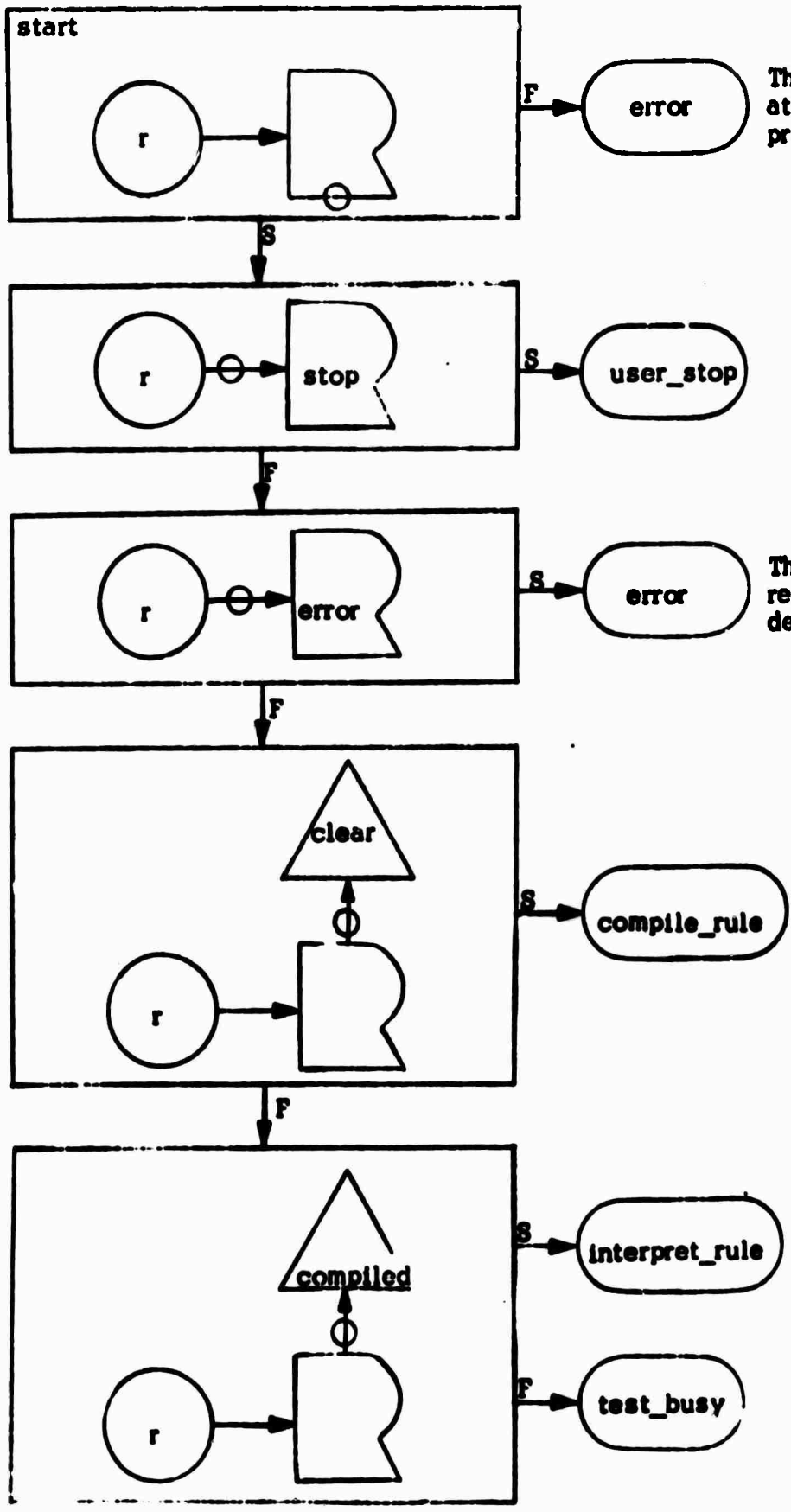


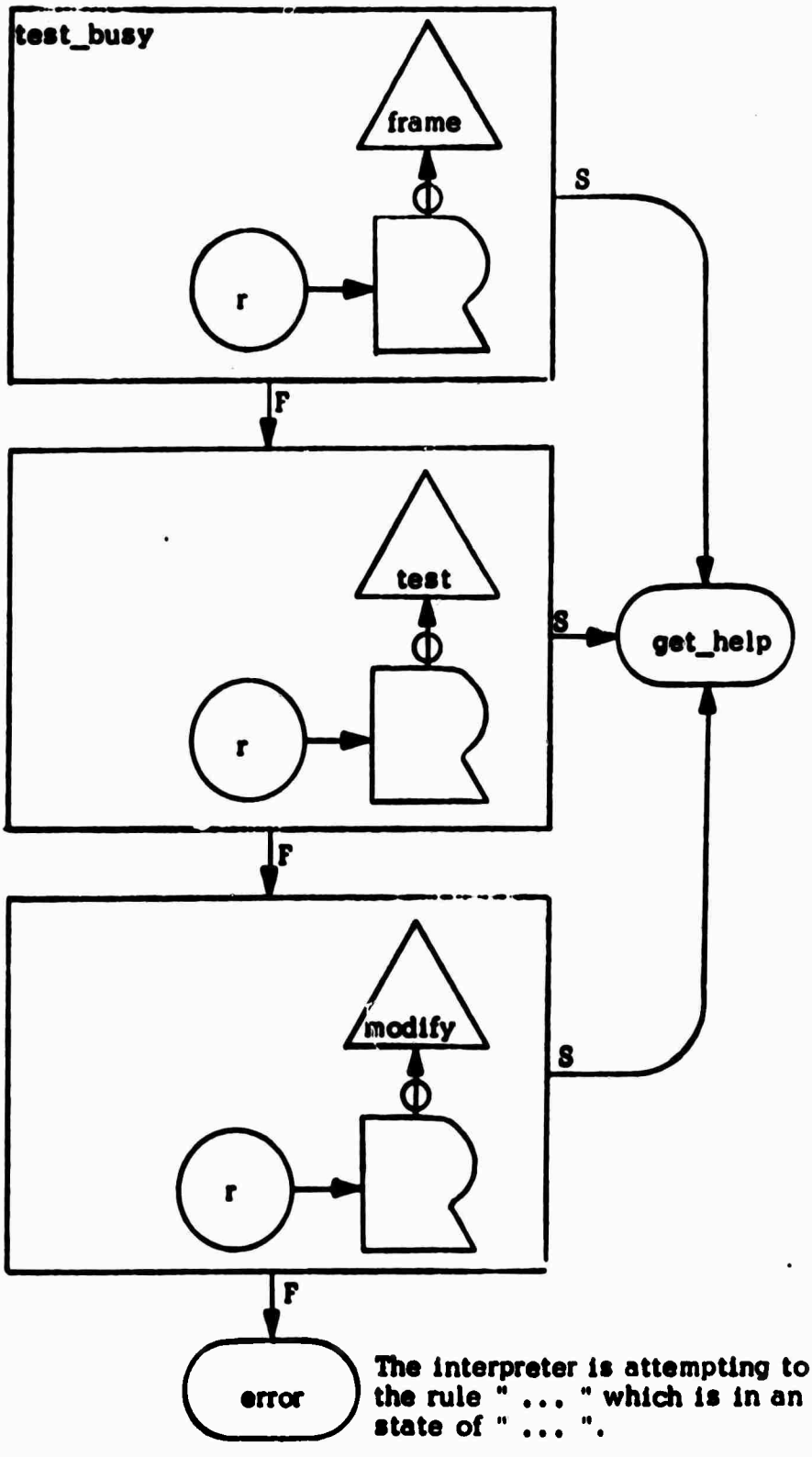
type :



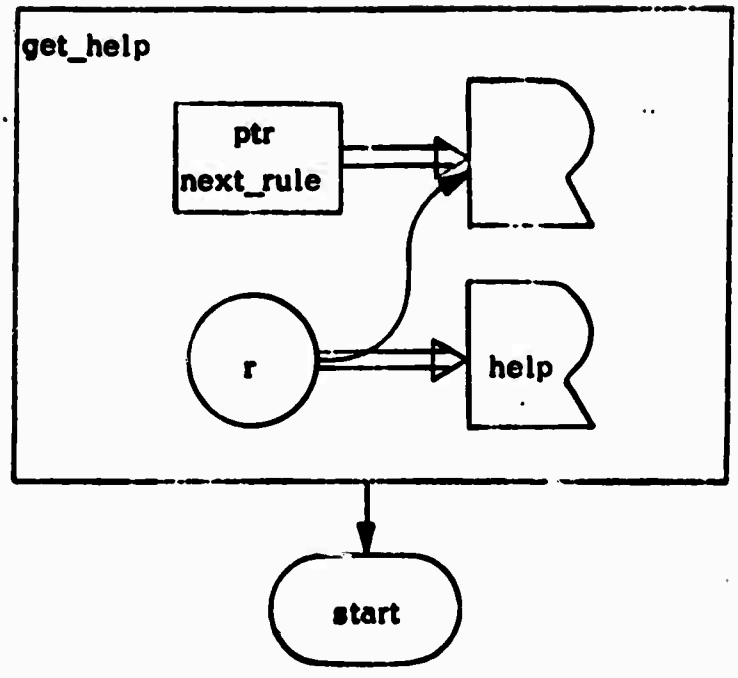
undef :

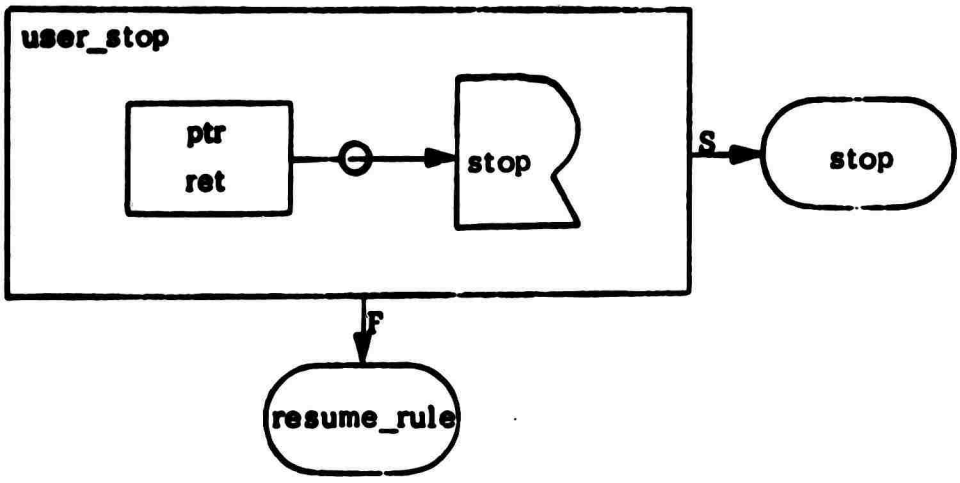


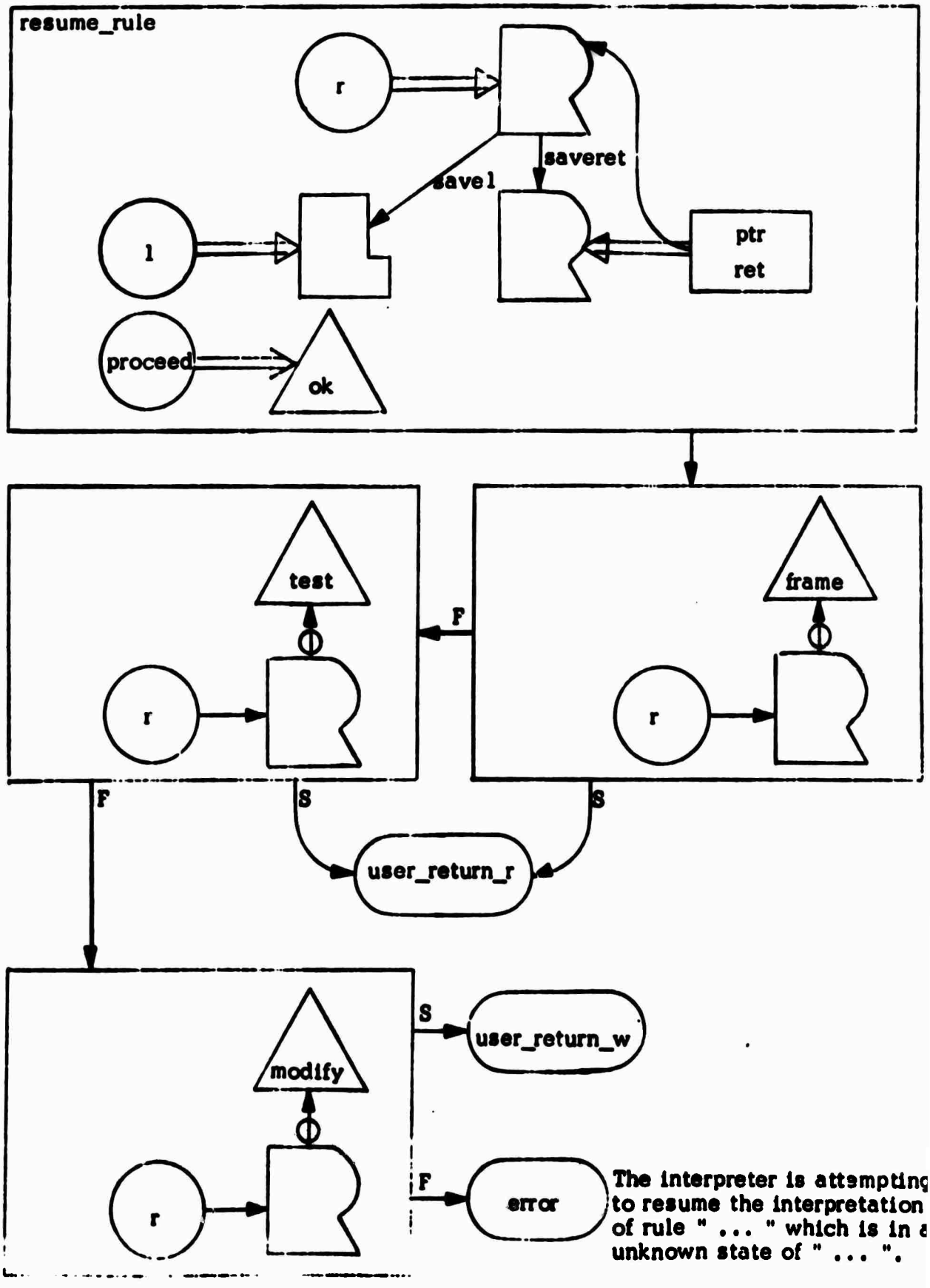

start
r
F
error
The interpreter is attempting to interpret a non-rule.
S
r
stop
S
user_stop
F
r
error
S
error
The interpreter is reporting a user-detected error.
F
clear
r
S
compile_rule
F
compiled
r
S
interpret_rule
F
test_busy

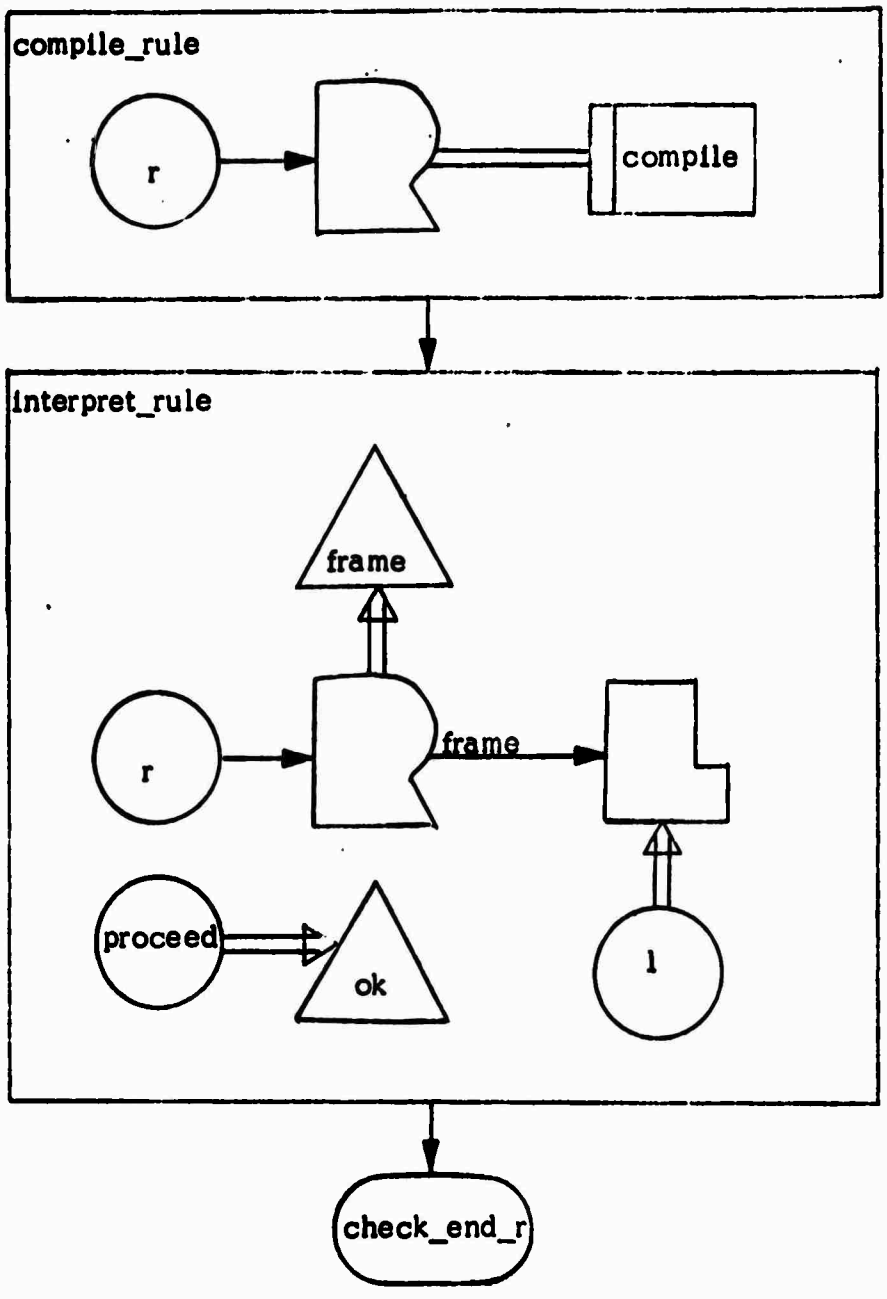


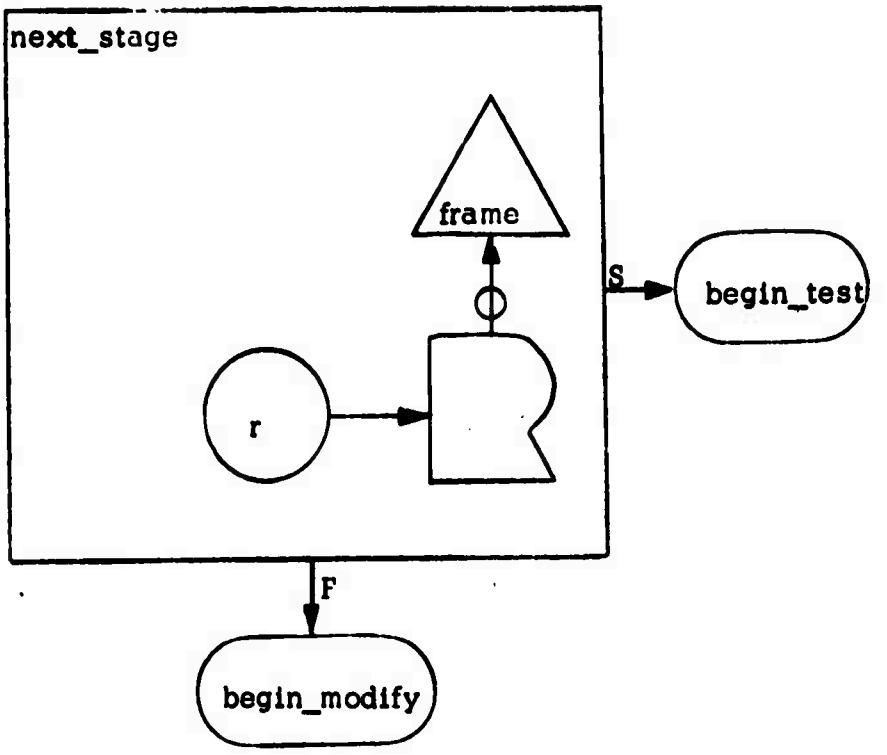
The interpreter is attempting to interpret the rule " ... " which is in an unknown state of " ... ".

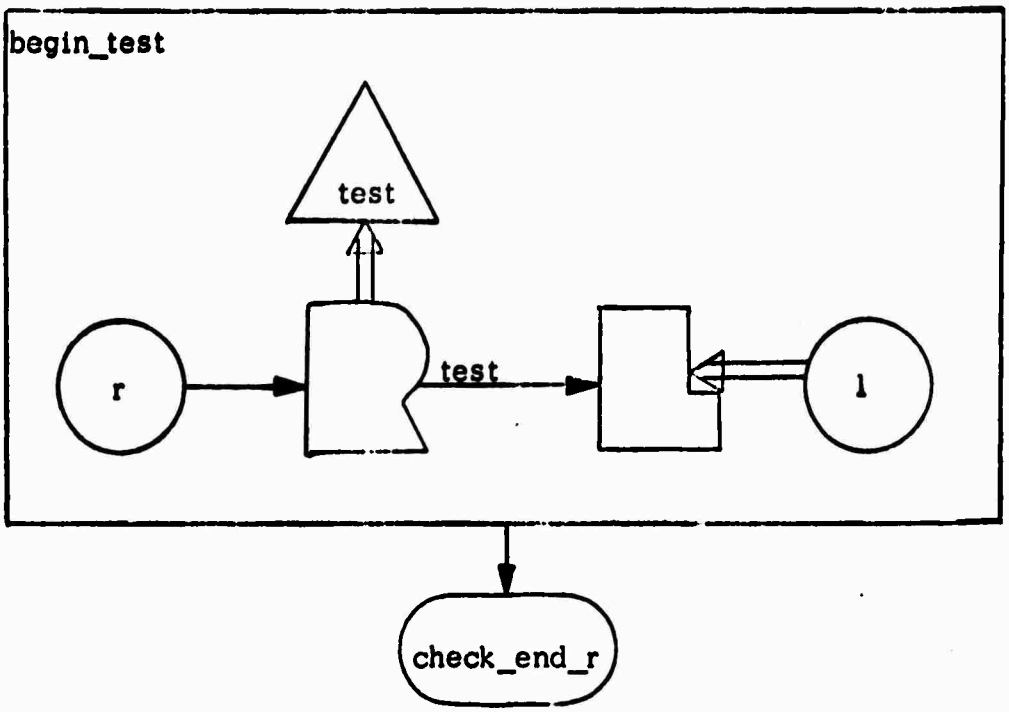

get_help
ptr
next_rule
r
help
start

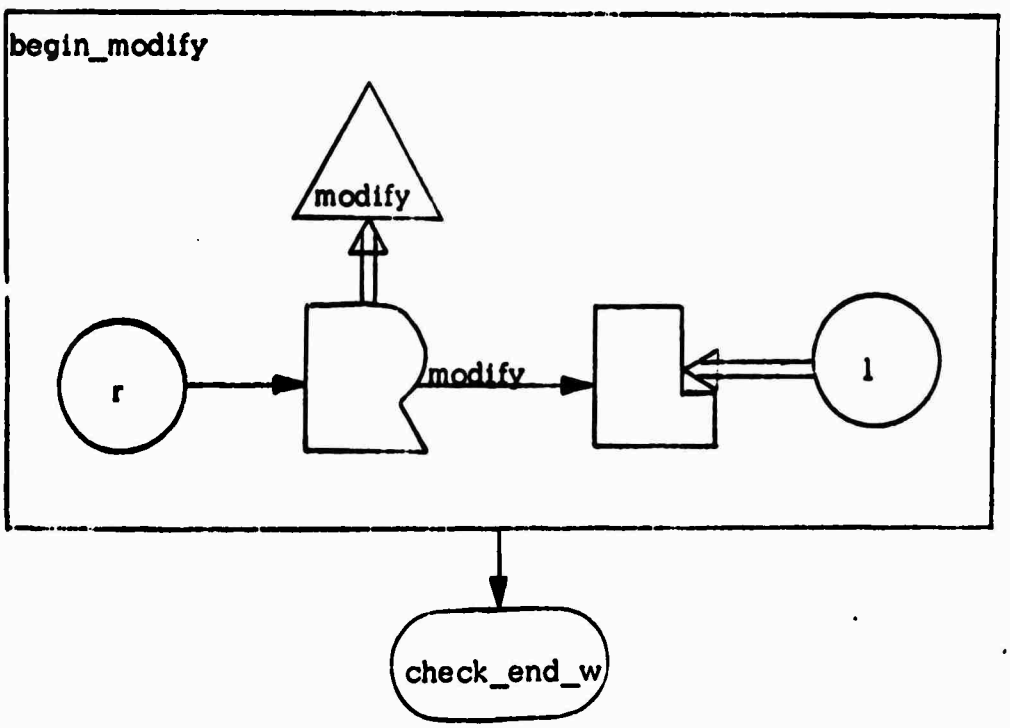

user_stop
ptr
ret
stop
S
stop
F
resume_rule

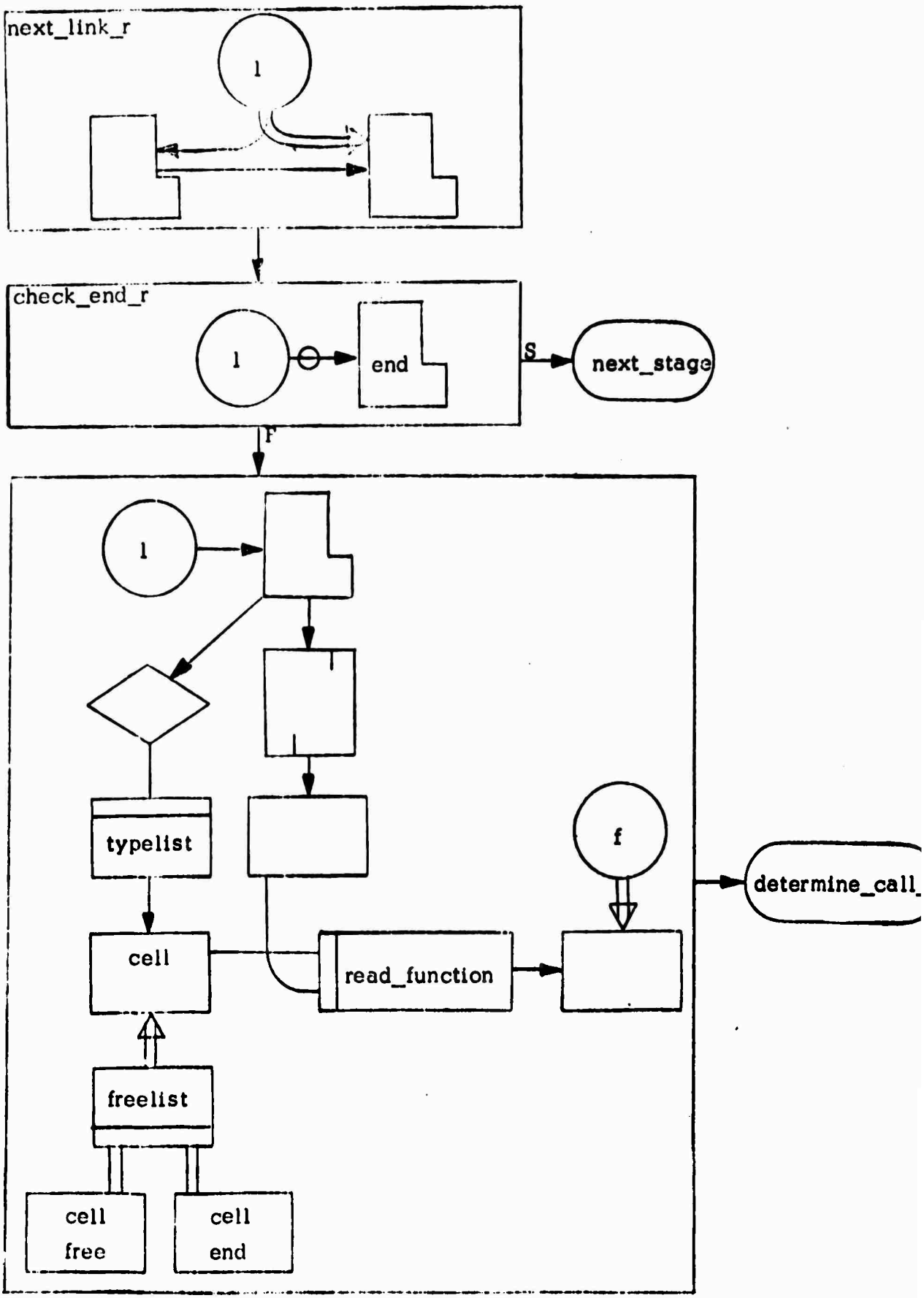

resume_rule
r
saveret
savel
l
ptr
ret
proceed
ok
test
F
frame
r
r
F
S
S
user_return_r
S
user_return_w
modify
r
F
error
The interpreter is attempting
to resume the interpretation
of rule " ... " which is in a
unknown state of " ... ".

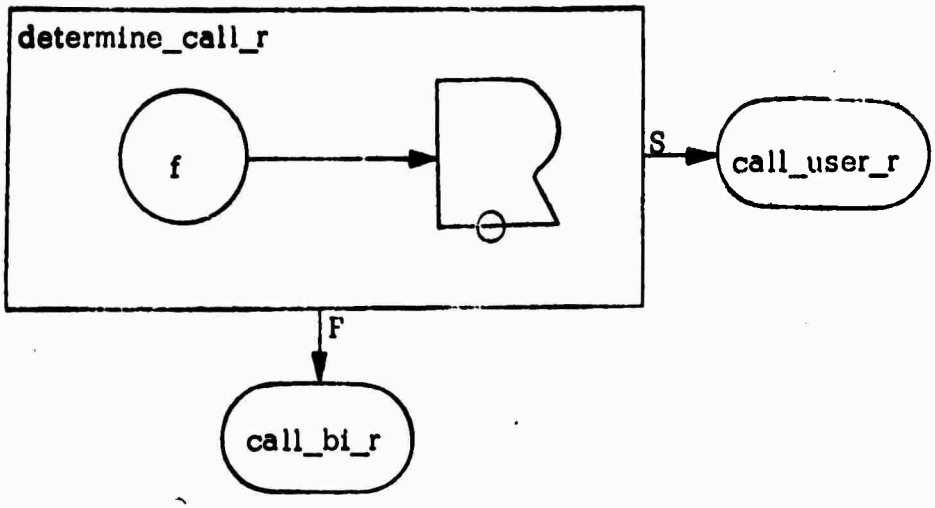

compile_rule
r
compile
interpret_rule
frame
r
frame
proceed
ok
l
check_end_r

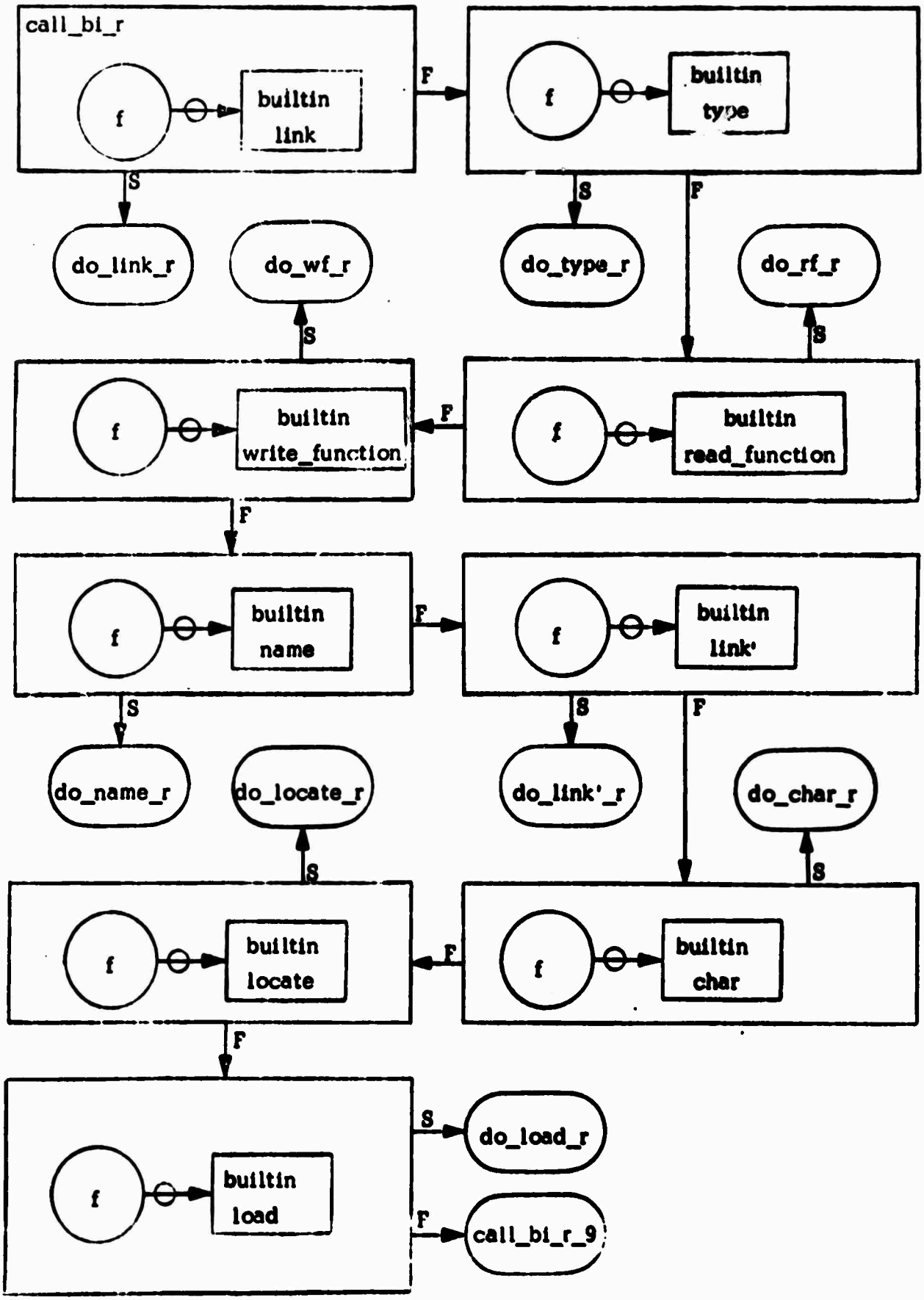

next_stage
frame
S
begin_test
r
F
begin_modify

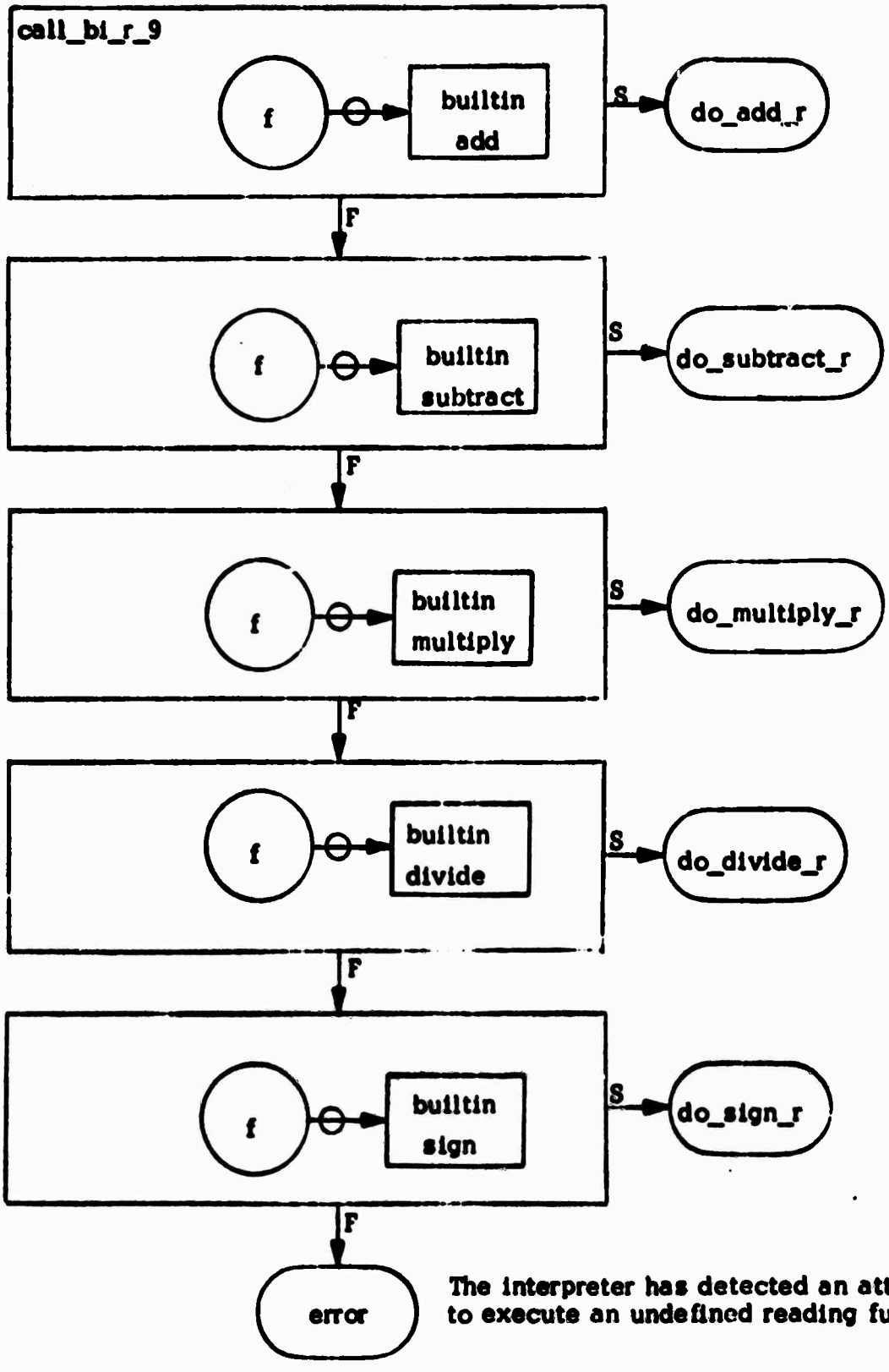

begin_test
test
r
test
1
check_end_r


begin_modify
modify
r
modify
1
check_end_w

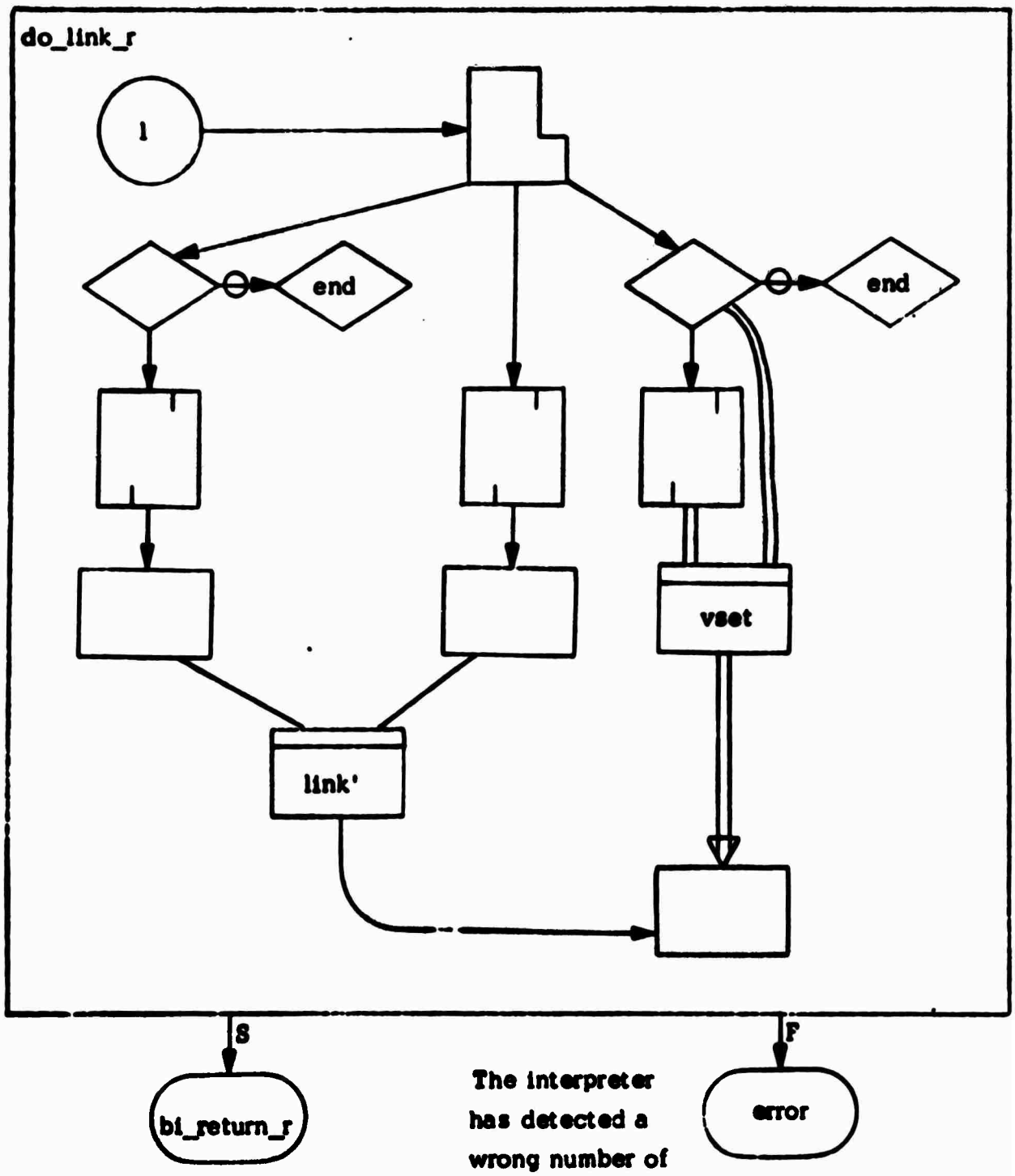

next_link_r
1
check_end_r
1
end
S
next_stage
F
1
typelist
f
determine_call_
cell
read_function
freelist
cell
free
cell
end


determine_call_r
f
S
call_user_r
F
call_bi_r

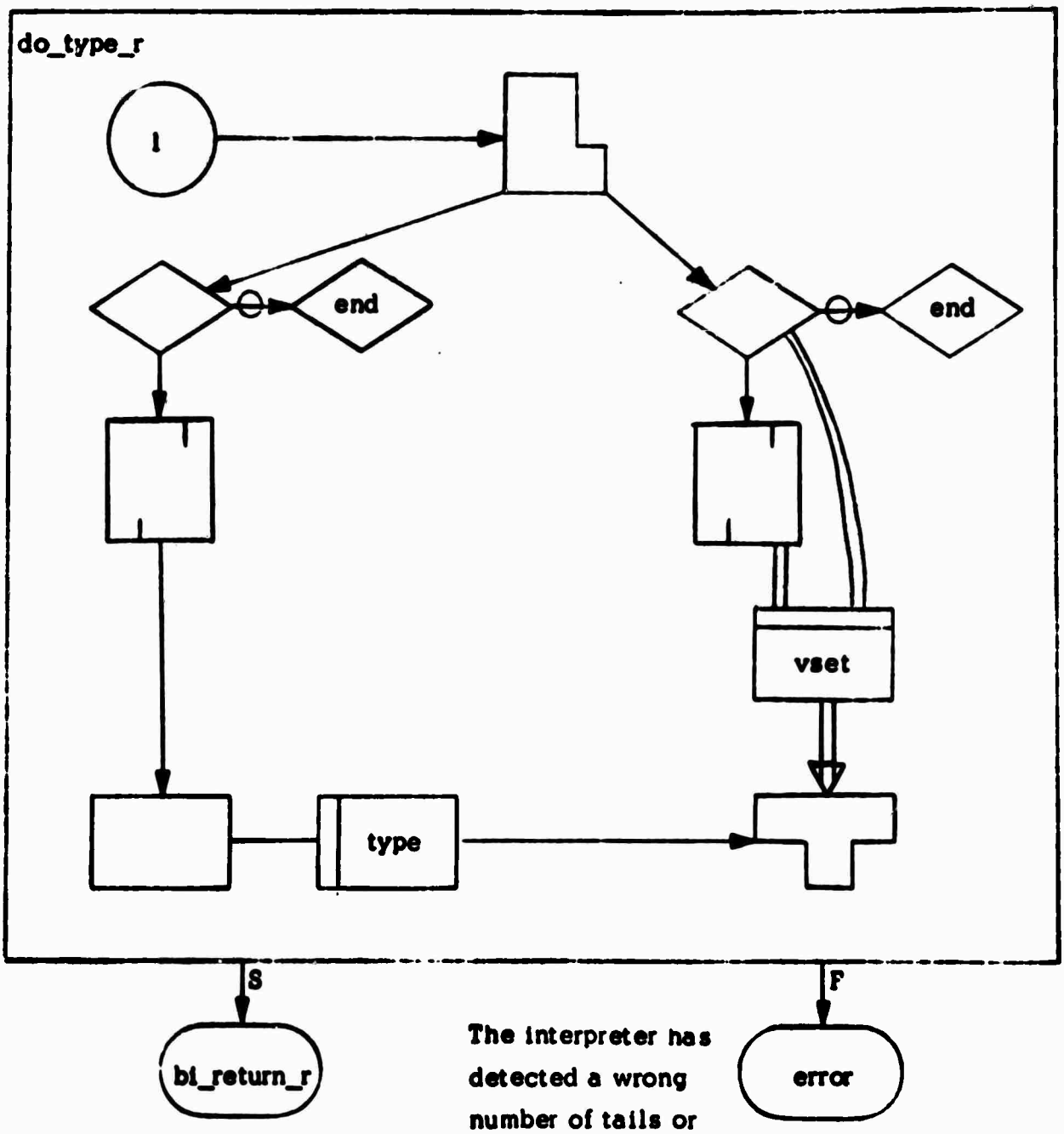

call_bi_r
f
builtin link
F
f
builtin type
S
S
F
do_link_r
do_wf_r
do_type_r
do_rf_r
S
S
f
builtin write_function
F
f
builtin read_function
F
f
builtin name
F
f
builtin link'
S
S
F
do_name_r
do_locate_r
do_link'_r
do_char_r
S
S
f
builtin locate
F
f
builtin char
F
f
builtin load
S
do_load_r
F
call_bi_r_9



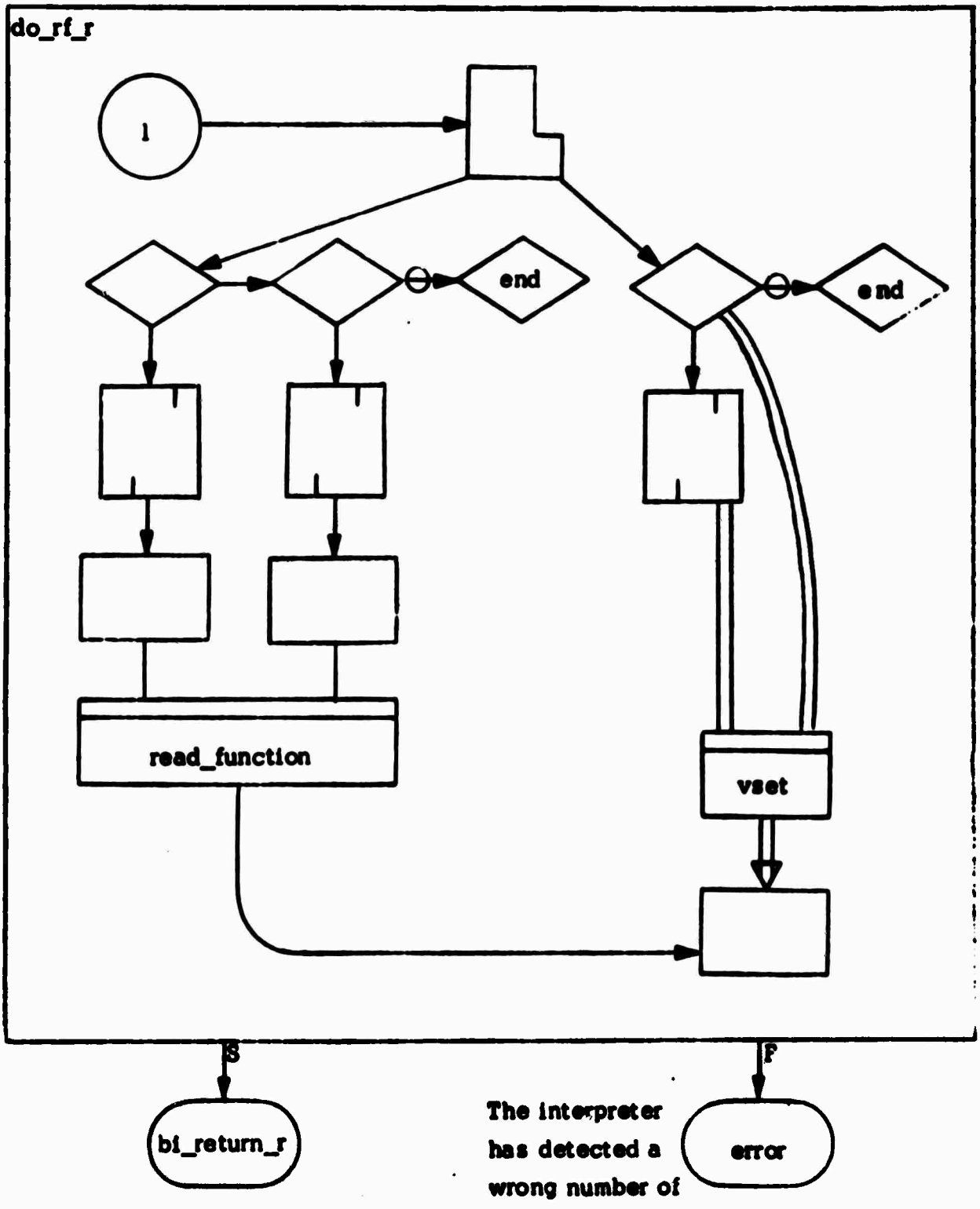
The interpreter has detected an attempt to execute an undefined reading function.



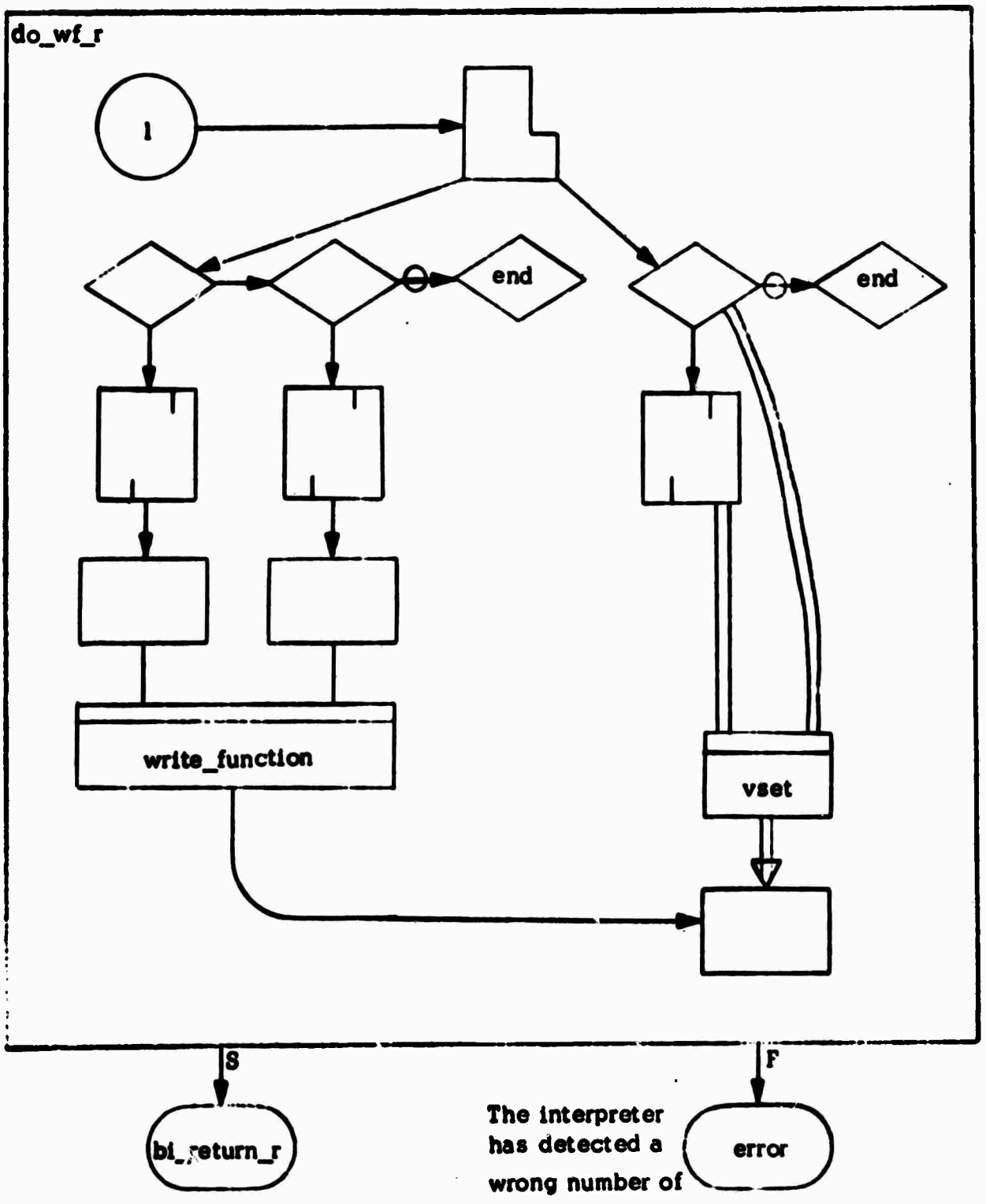
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "link".



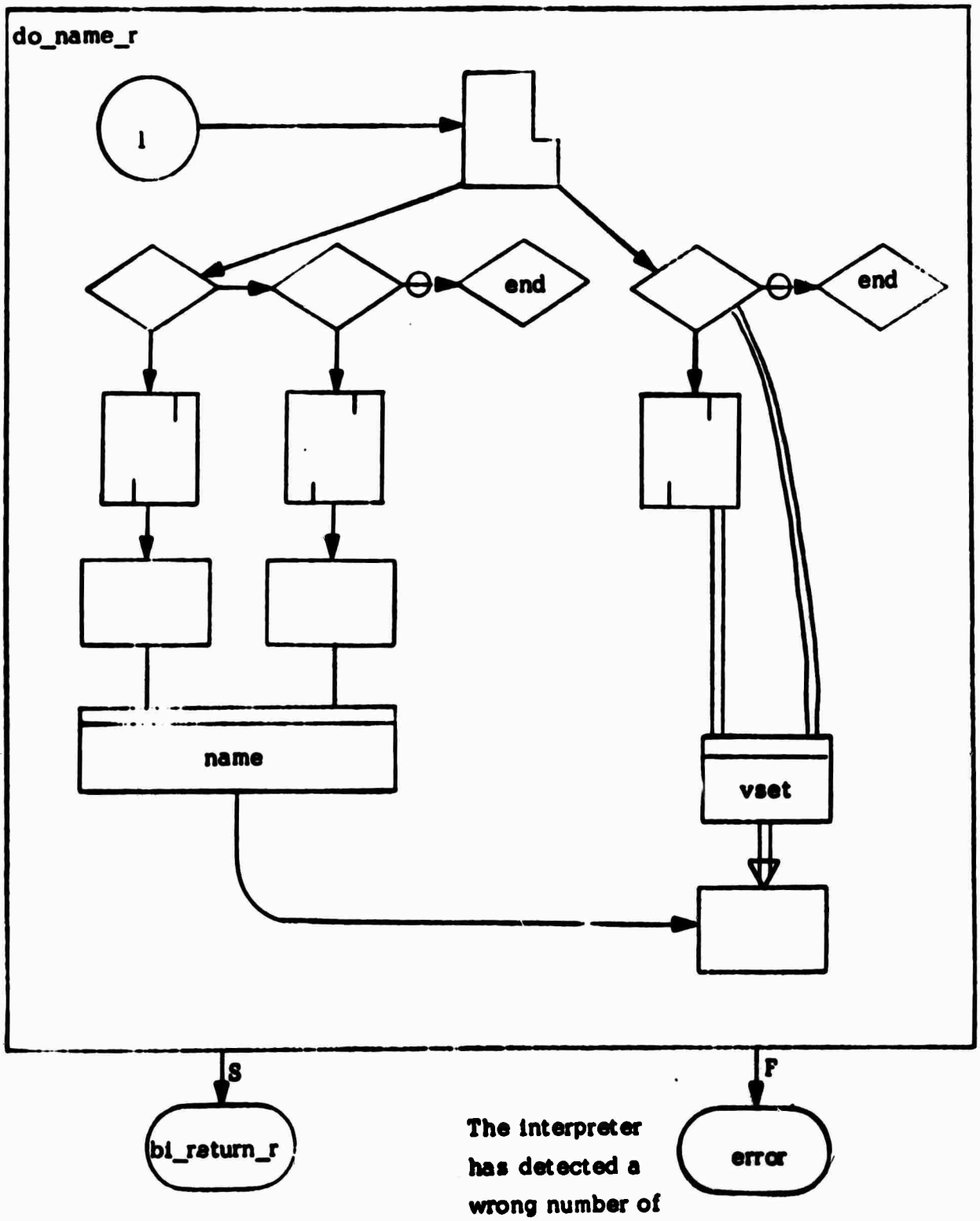
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "type".



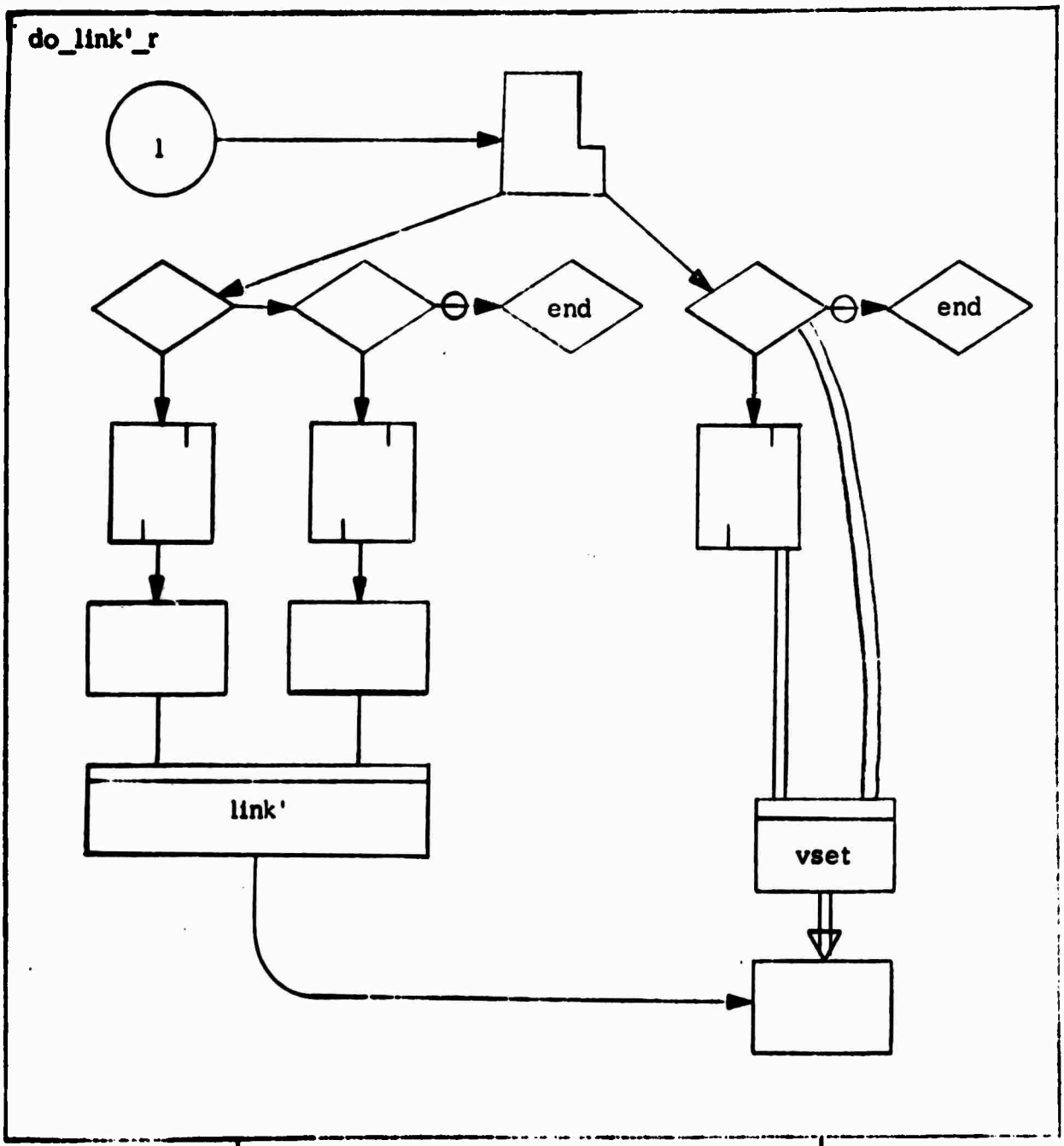
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "read_function".



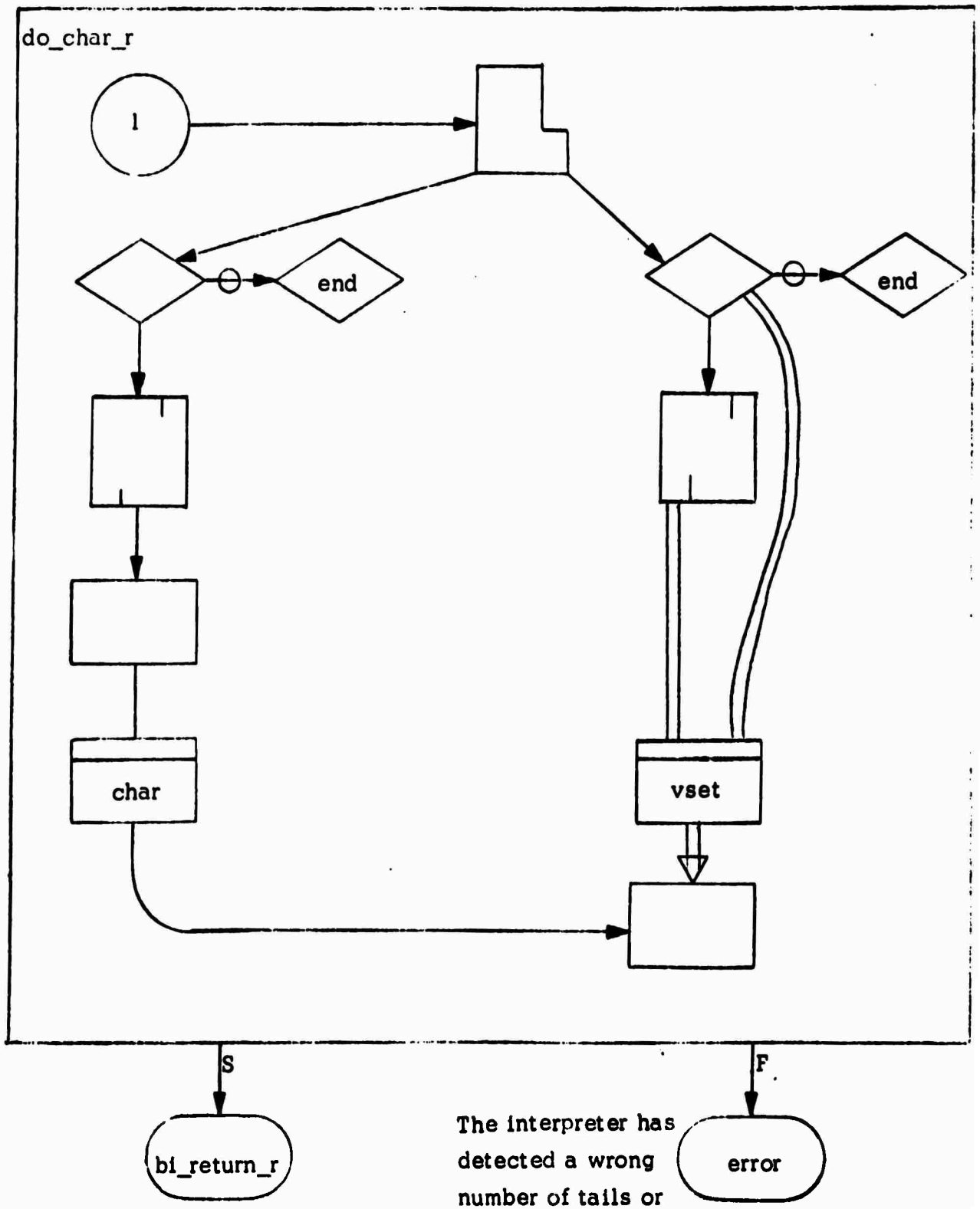
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "write_function".



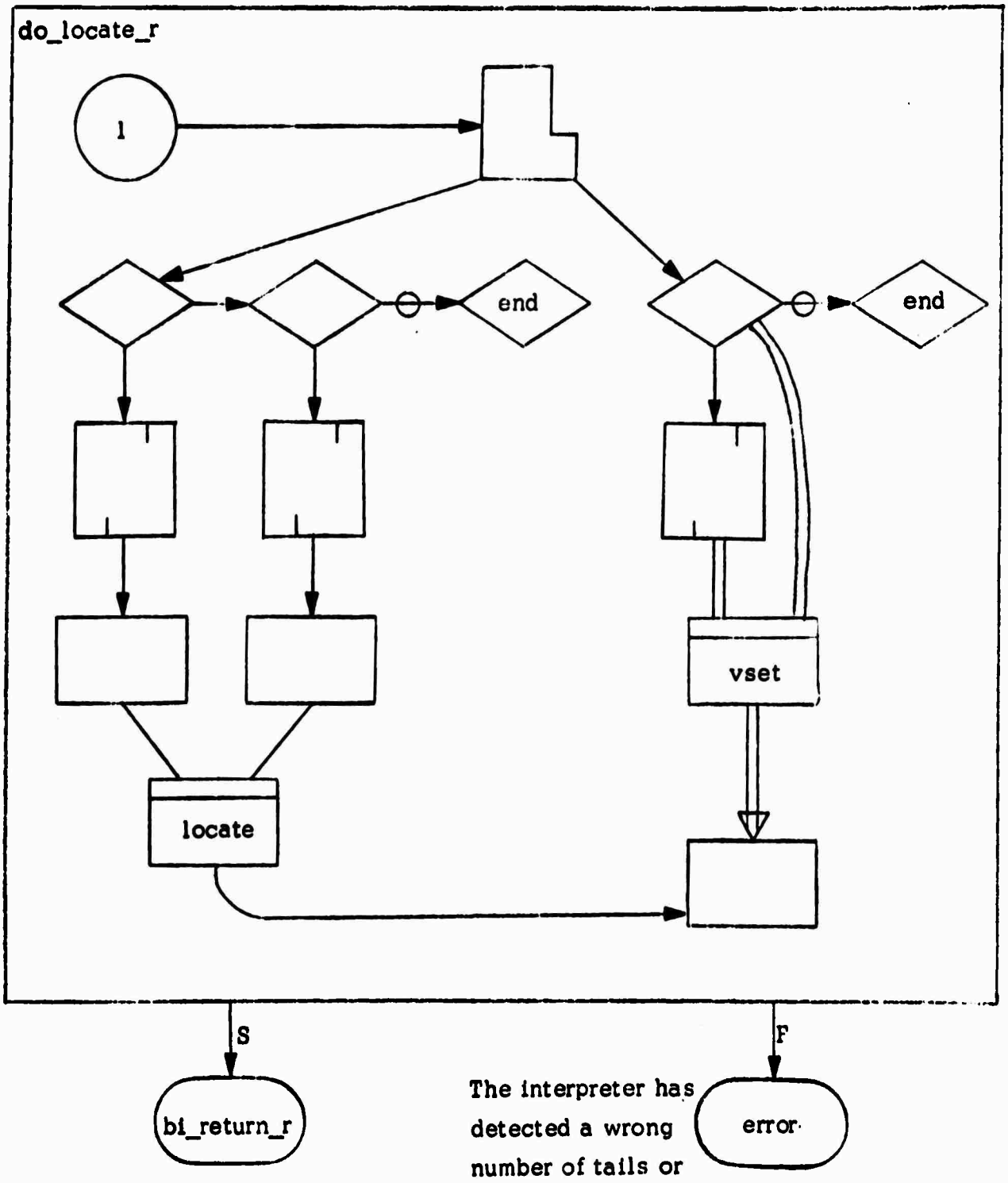
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "name".



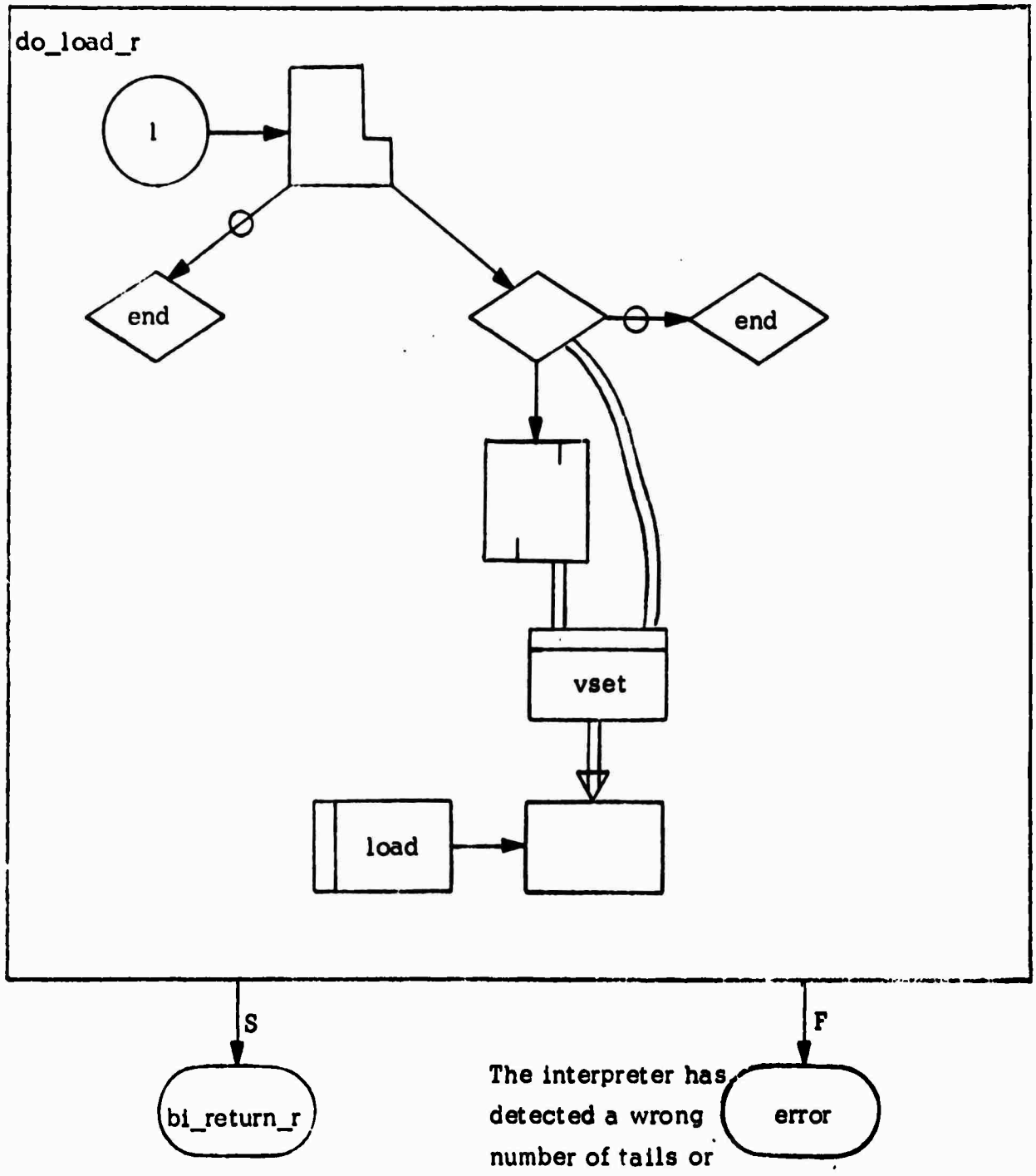
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "link'".



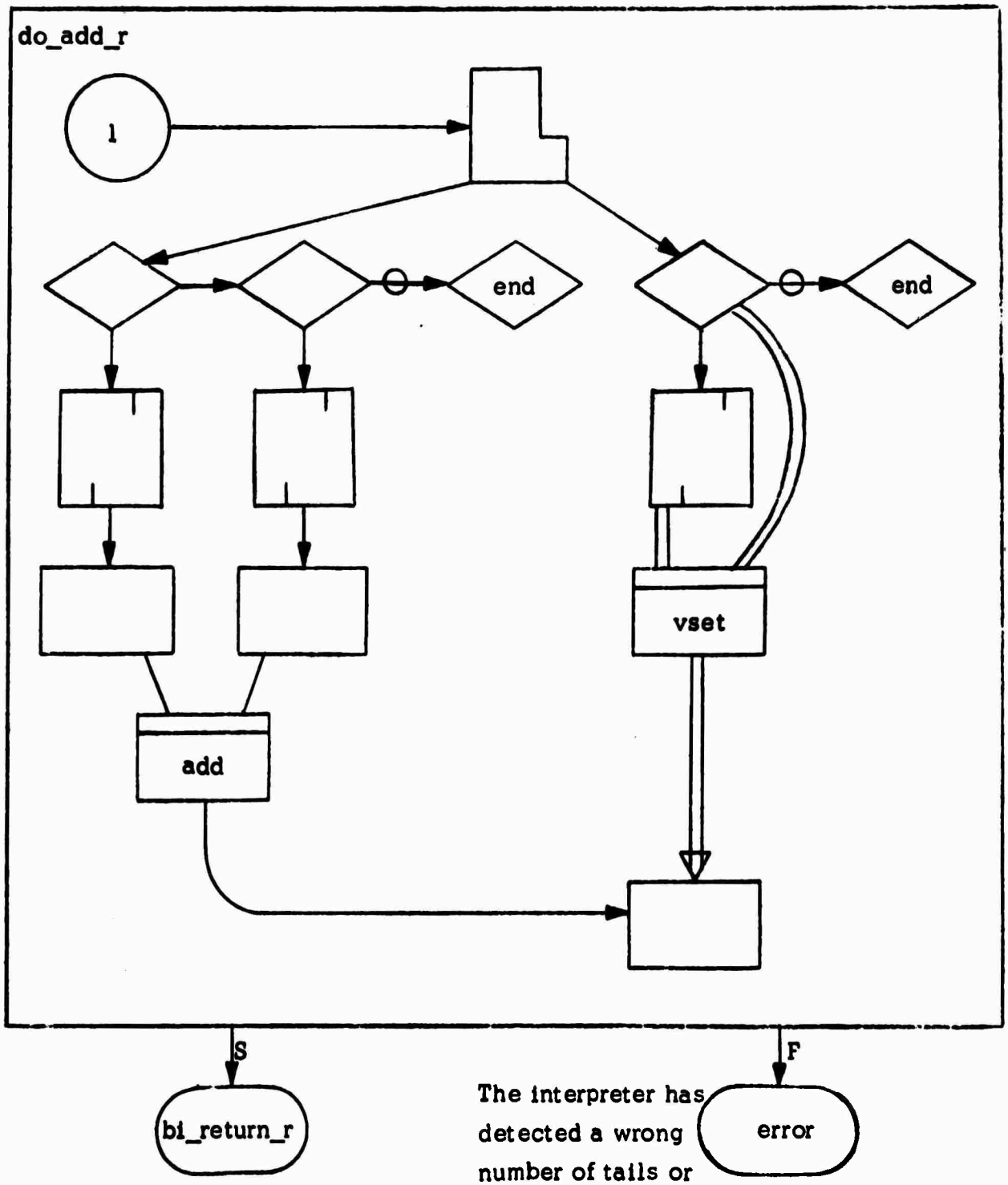
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "char".



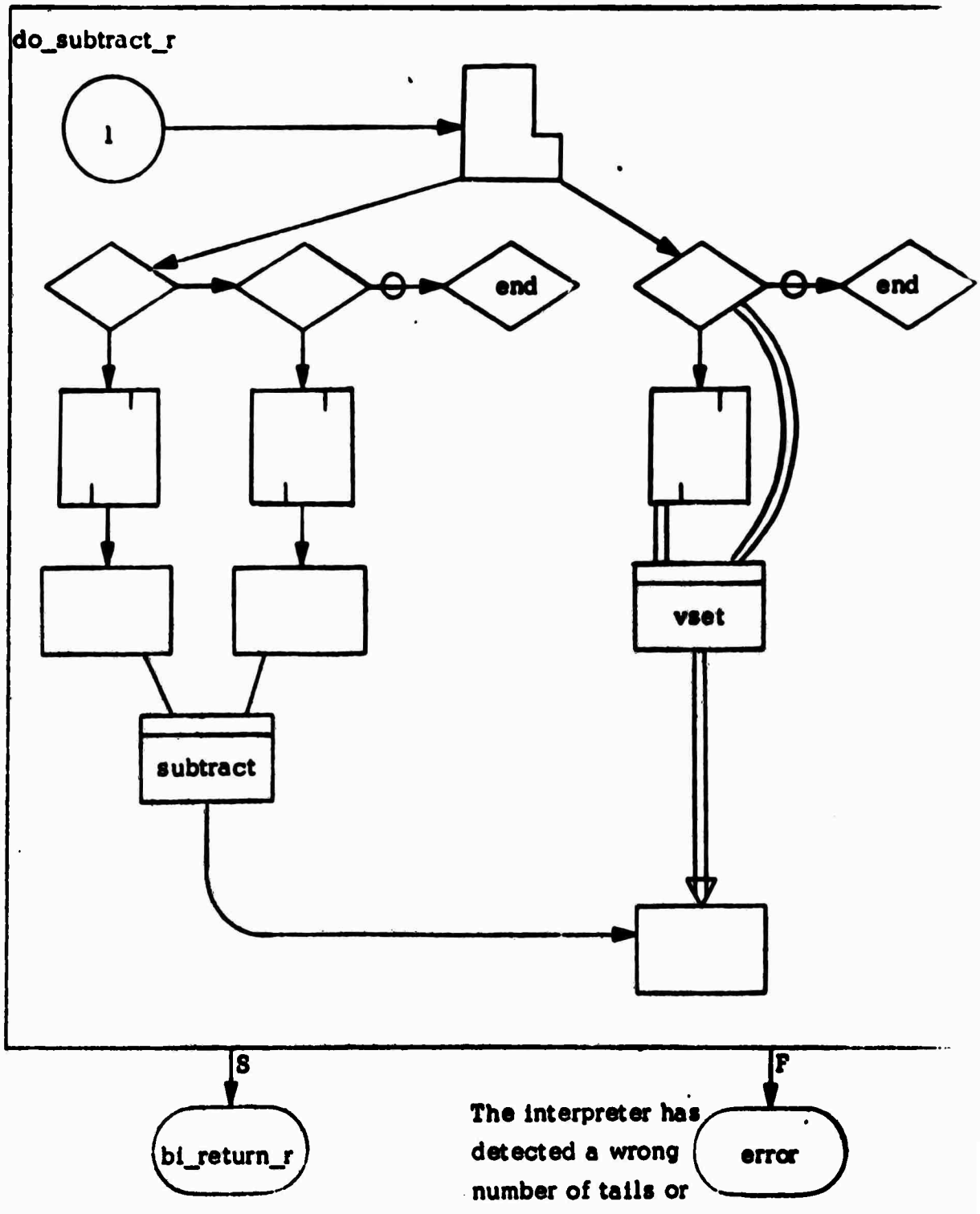
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "locate".



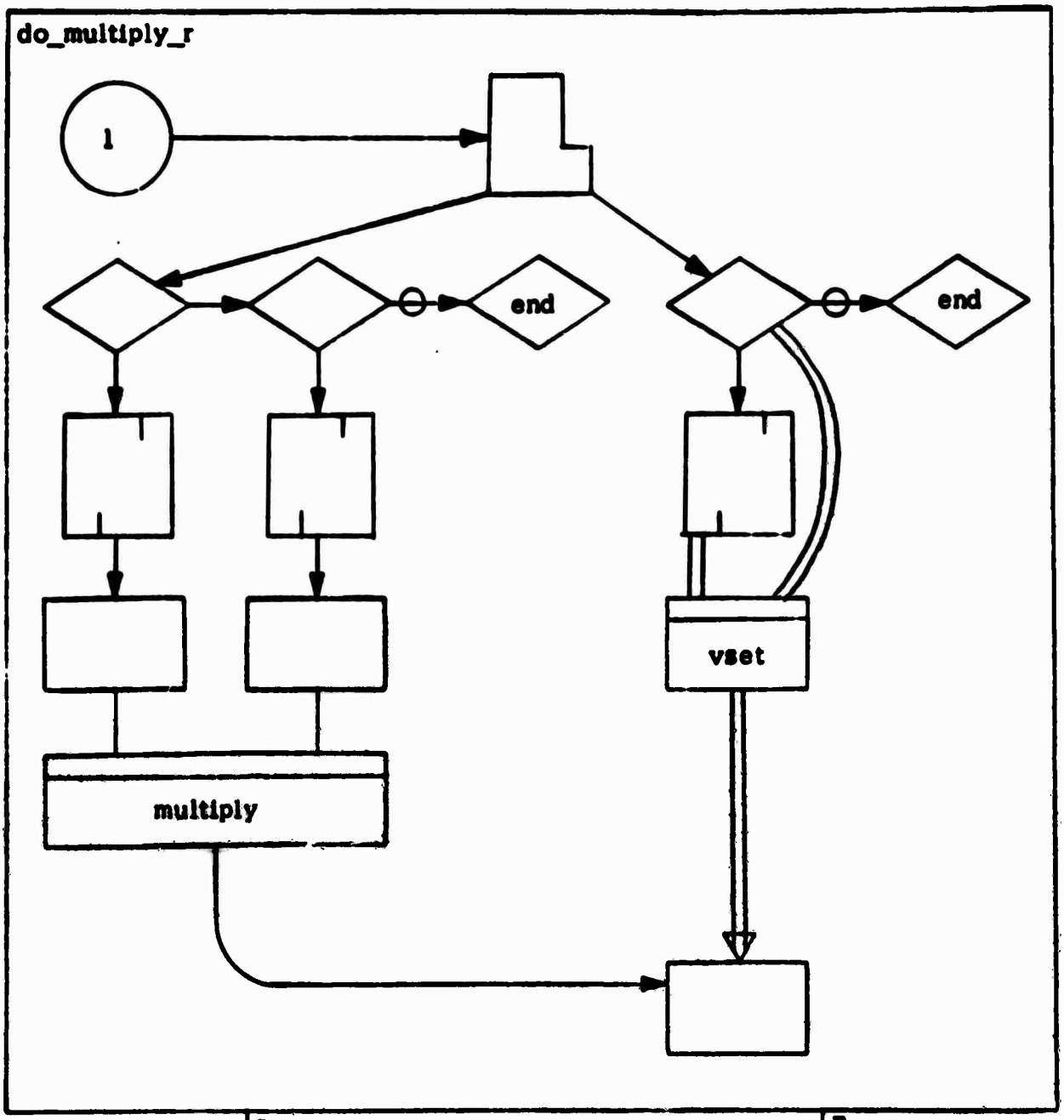
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "load".



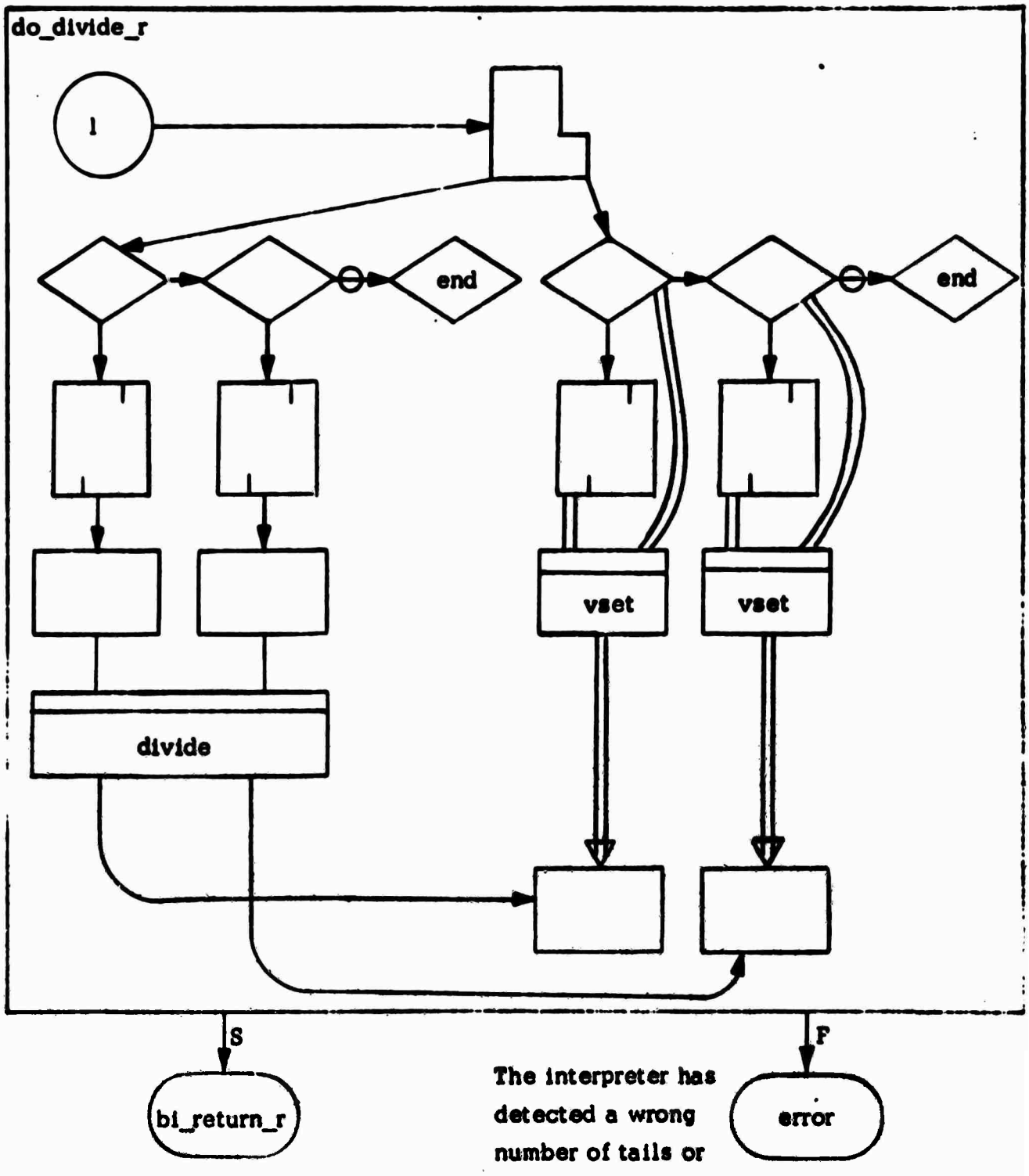
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "add".



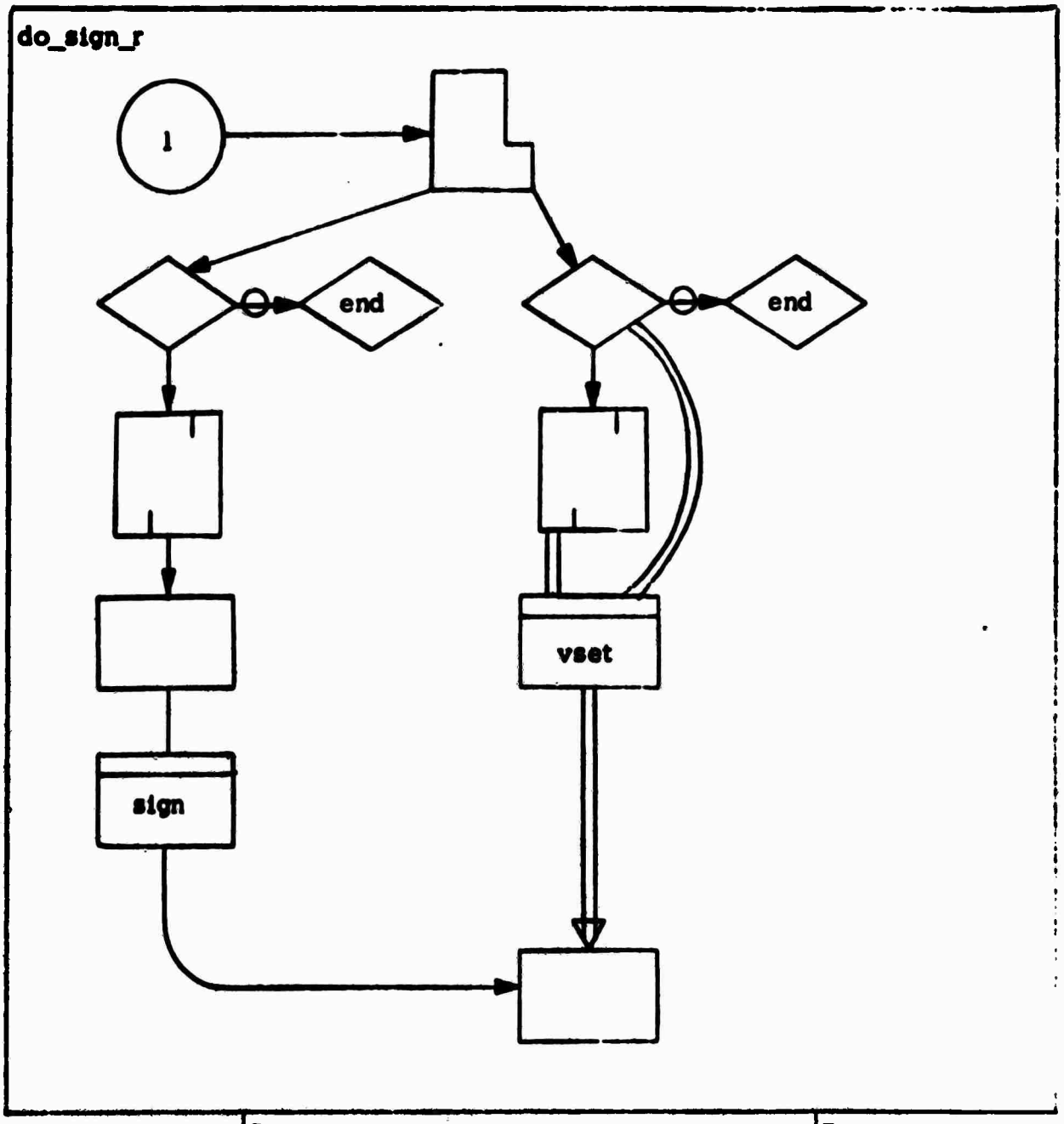
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "subtract".



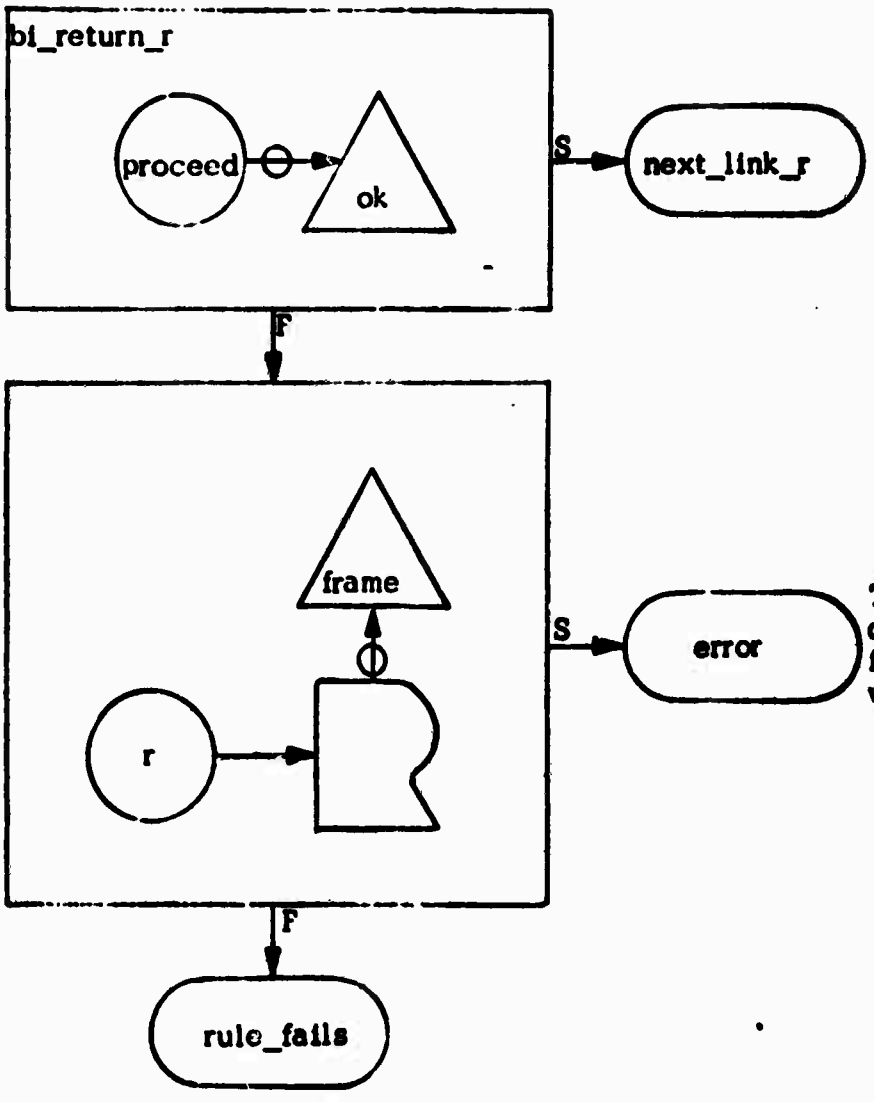
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "multiply".



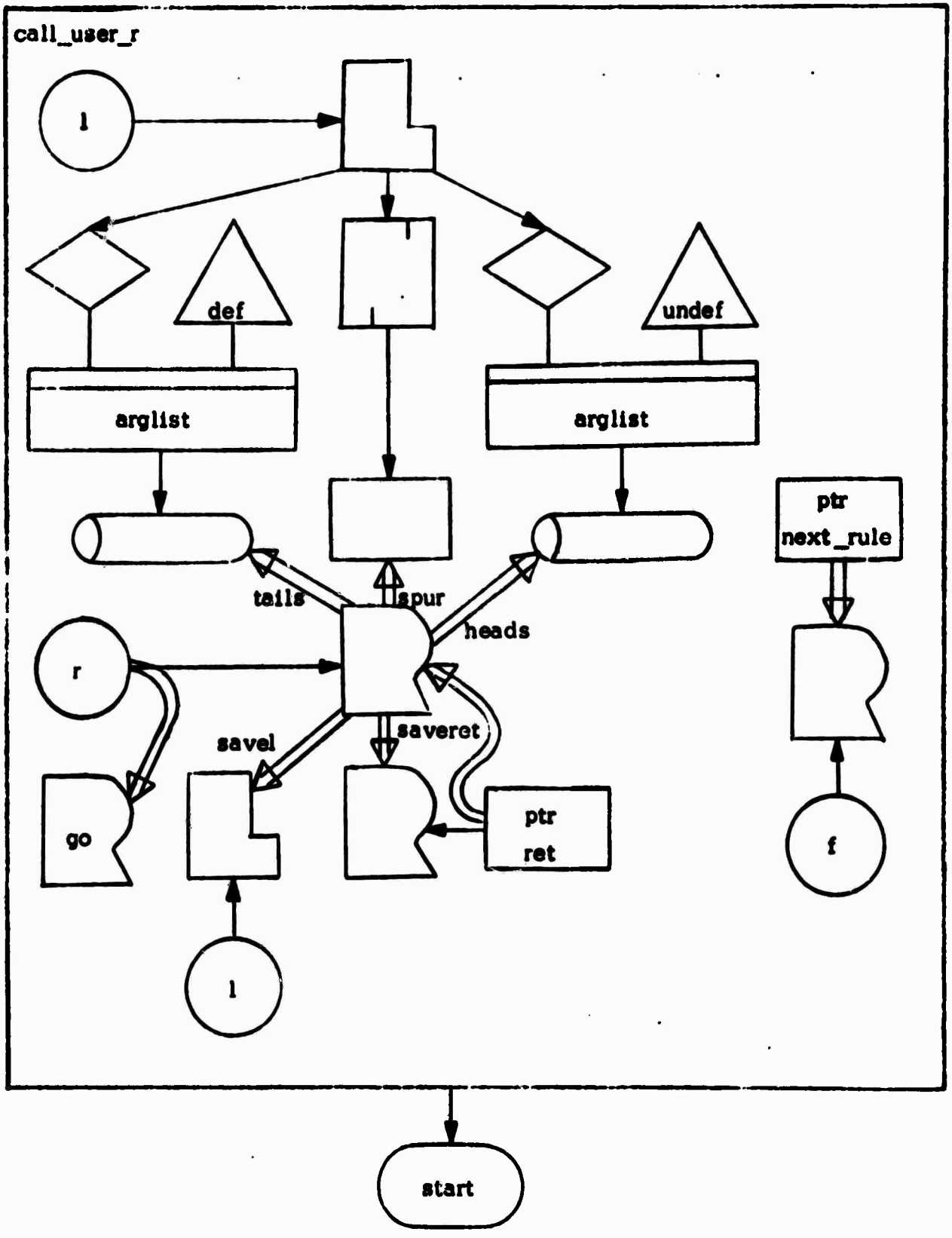
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "divide".

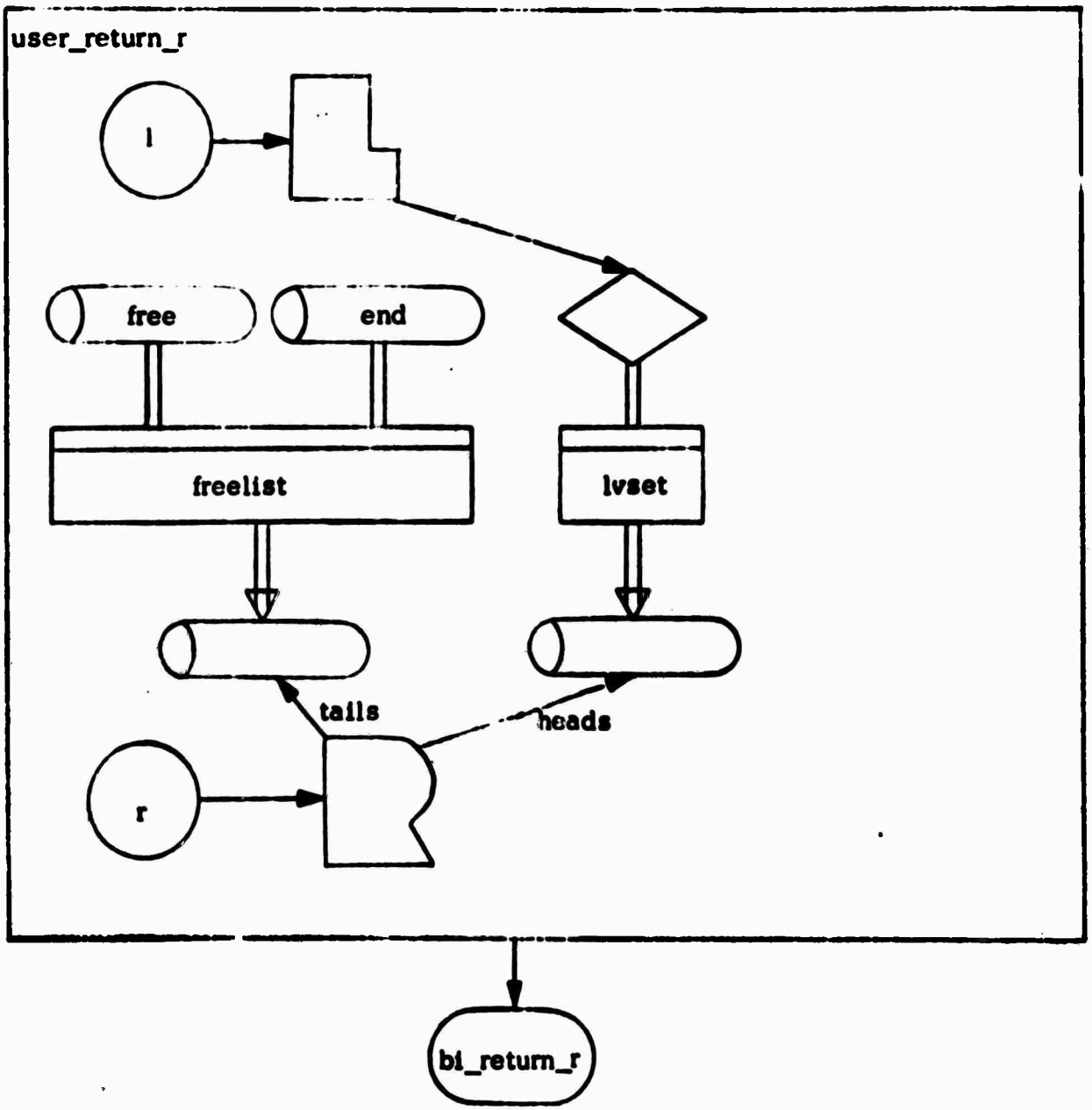


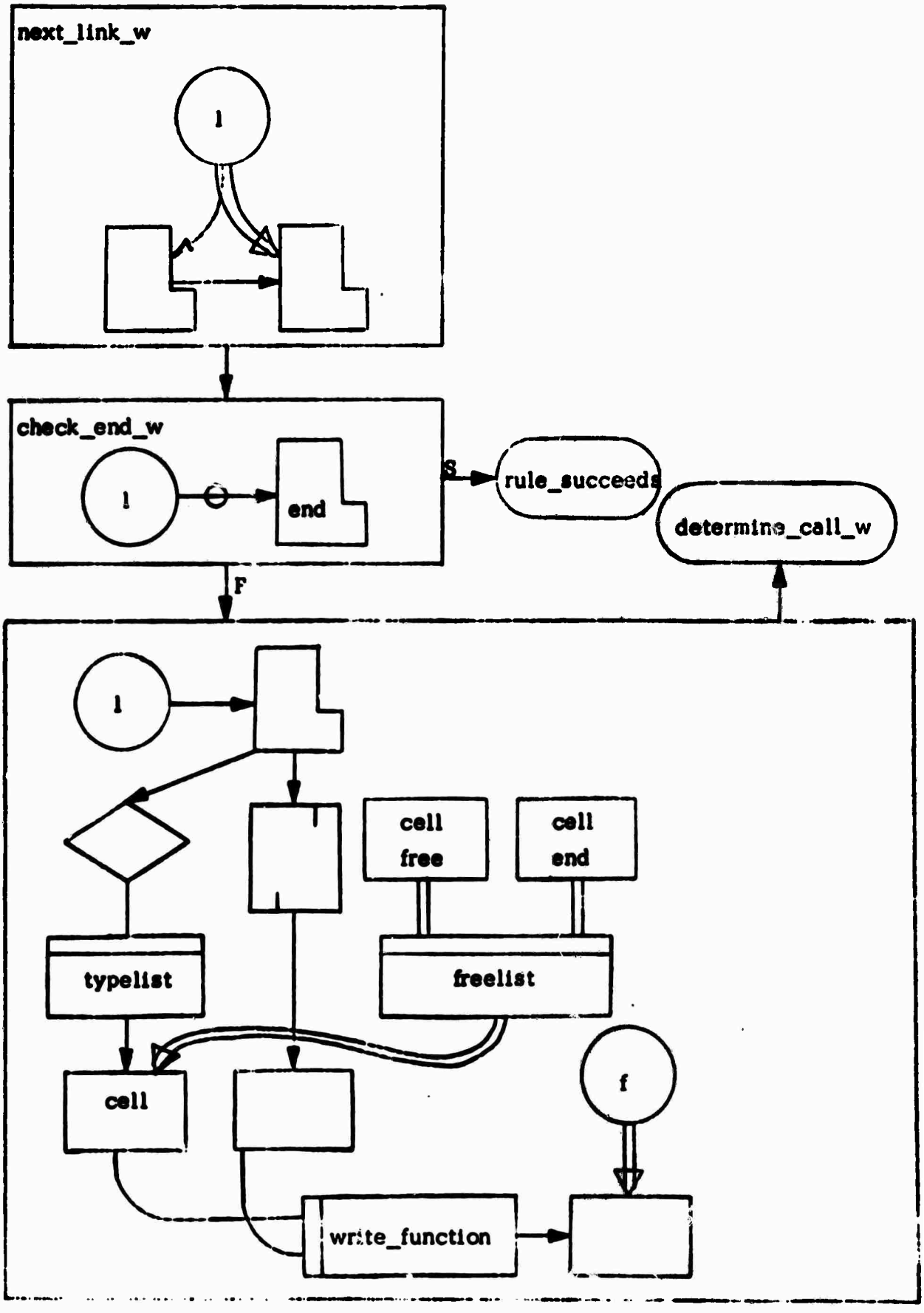
The interpreter has detected a wrong number of tails or heads on a read-call on the builtin "sign".

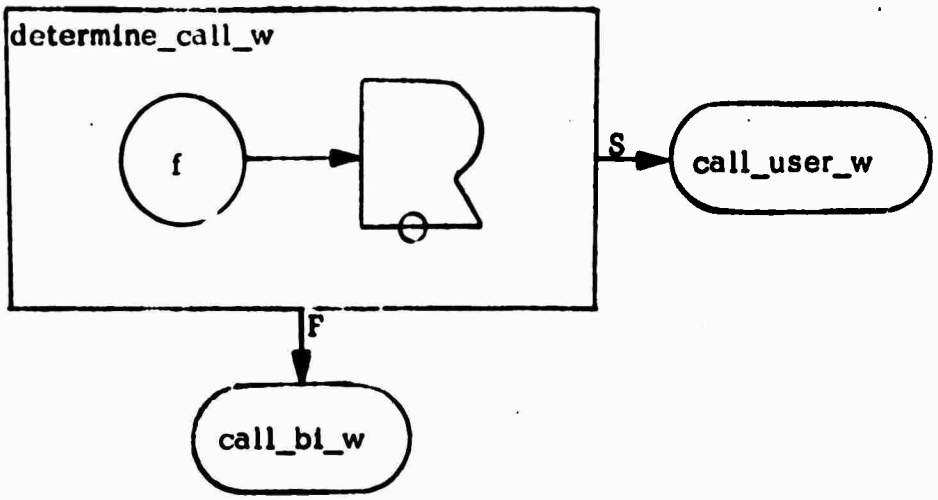


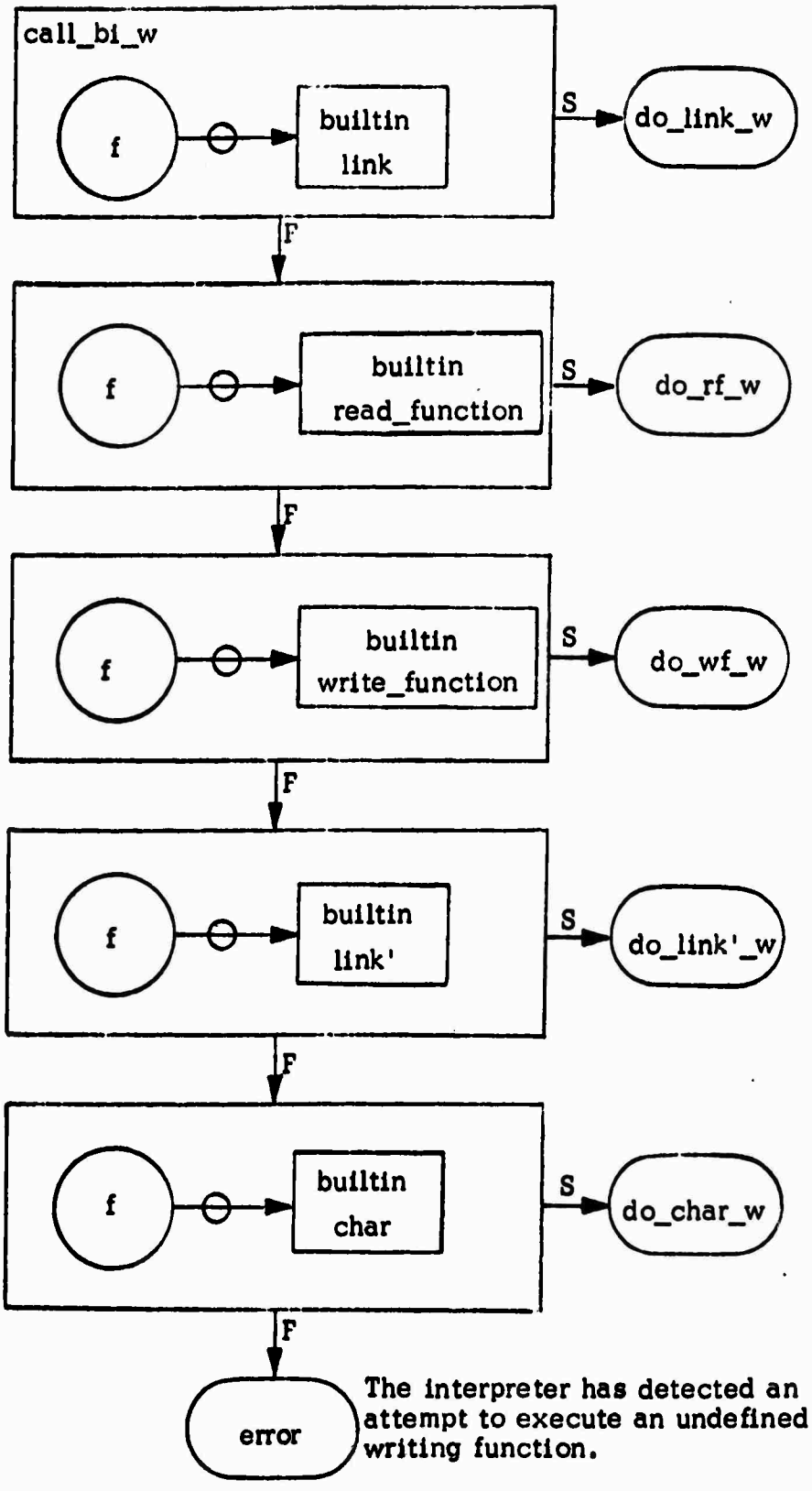
The interpreter has detected that the frame is inconsistent with the data graph.

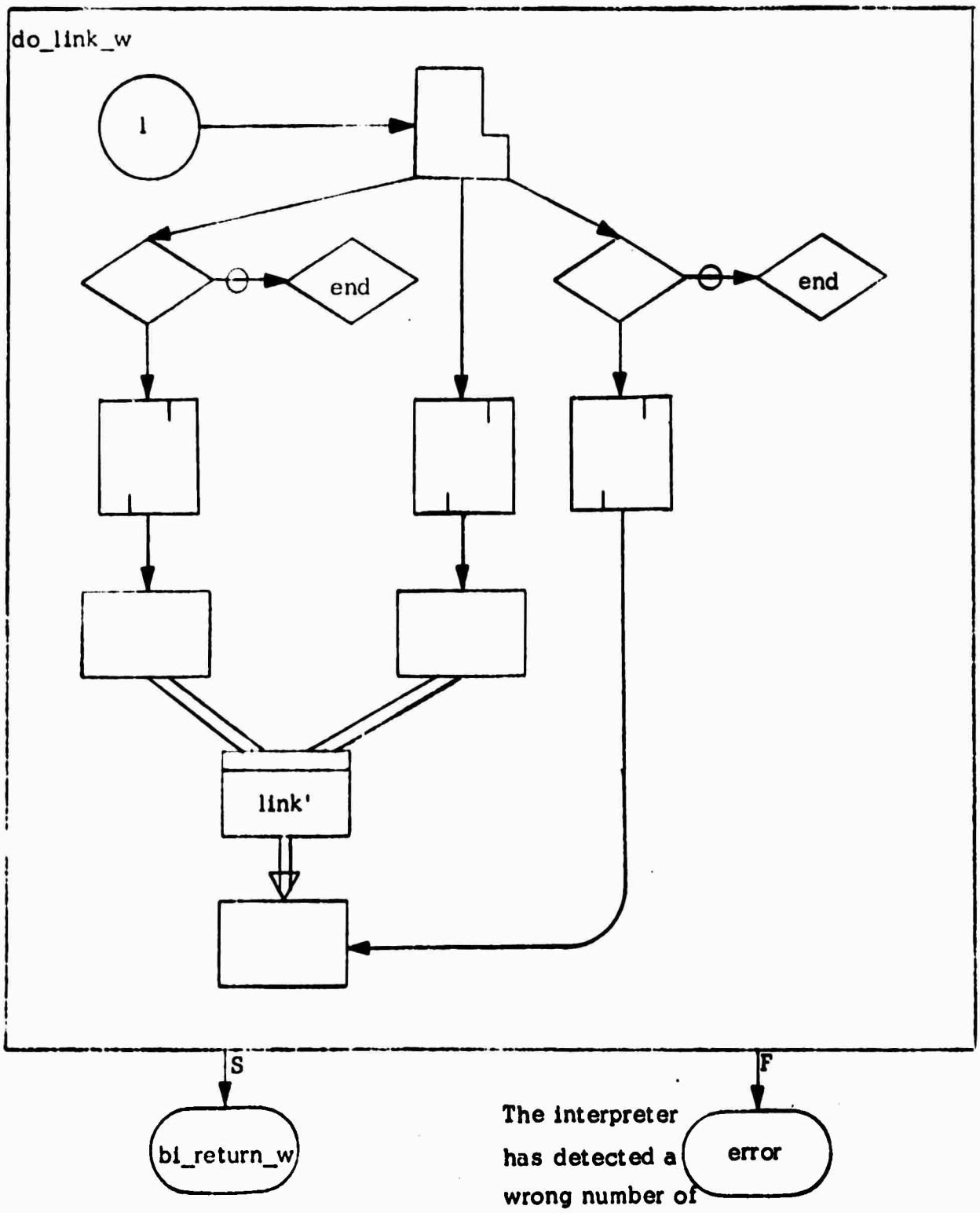

call_user_r
l
def
undef
arglist
arglist
ptr
next_rule
tails
spur
heads
r
savel
saveret
go
ptr
ret
f
l
start


user_return_r
l
free
end
freelist
lvset
tails
heads
r
bi_return_r

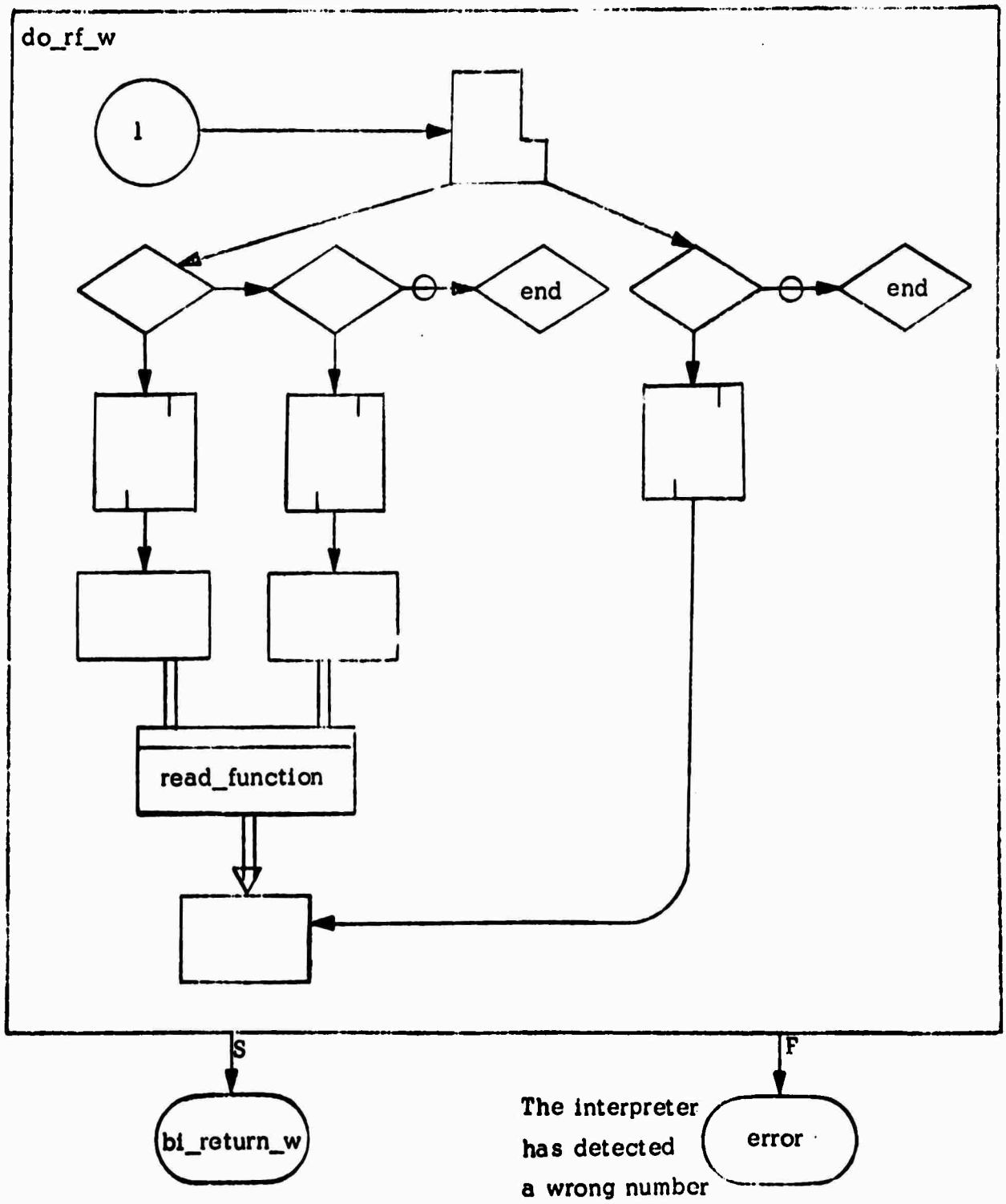

next_link_w
1
check_end_w
1
end
S
rule_succeeds
determine_call_w
F
1
cell
free
cell
end
typelist
freelist
cell
f
write_function


determine_call_w
f
S
call_user_w
F
call_bi_w

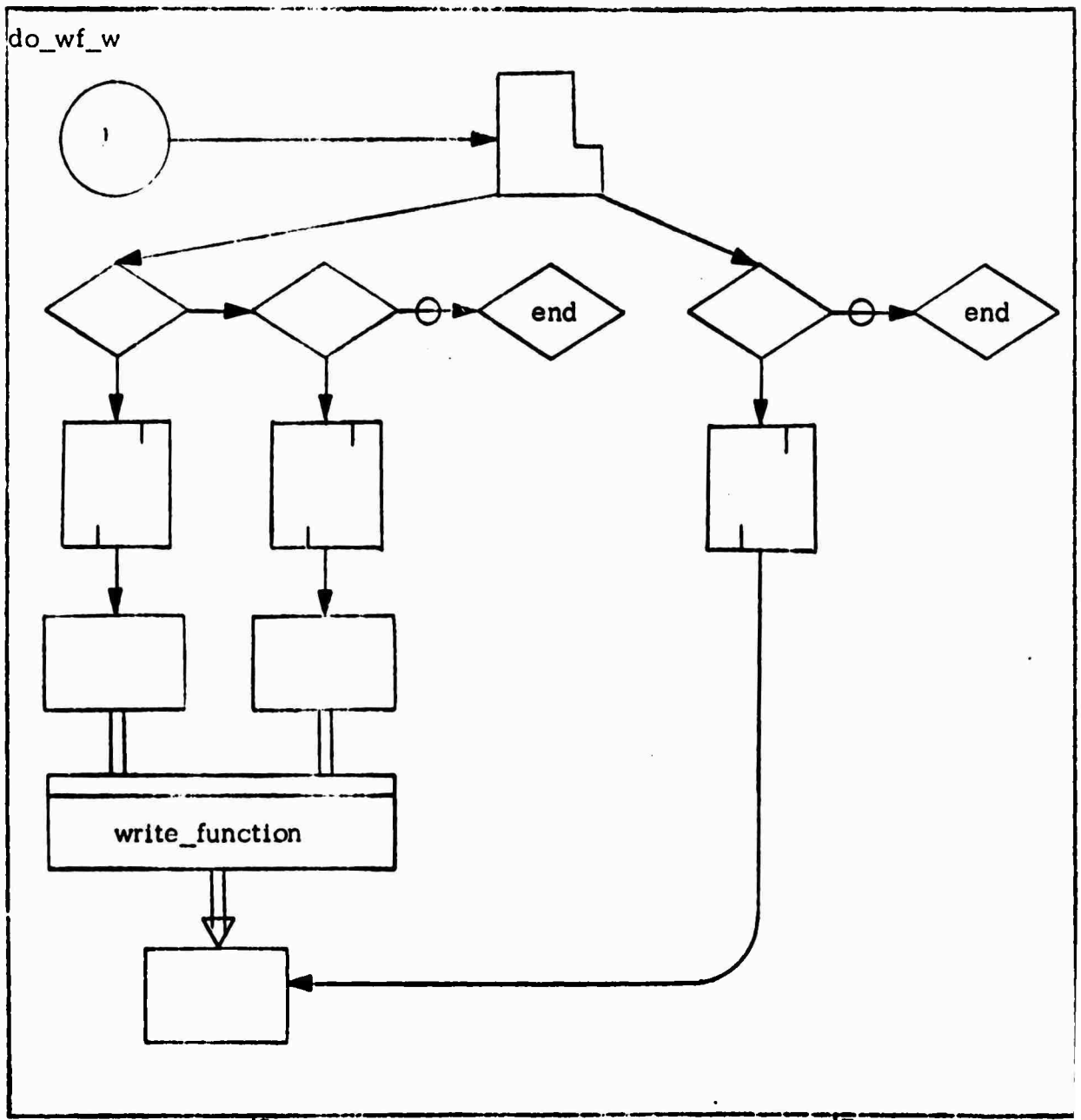

call_bi_w
f
builtin
link
S
do_link_w
F
f
builtin
read_function
S
do_rf_w
F
f
builtin
write_function
S
do_wf_w
F
f
builtin
link'
S
do_link'_w
F
f
builtin
char
S
do_char_w
F
error
The interpreter has detected an attempt to execute an undefined writing function.



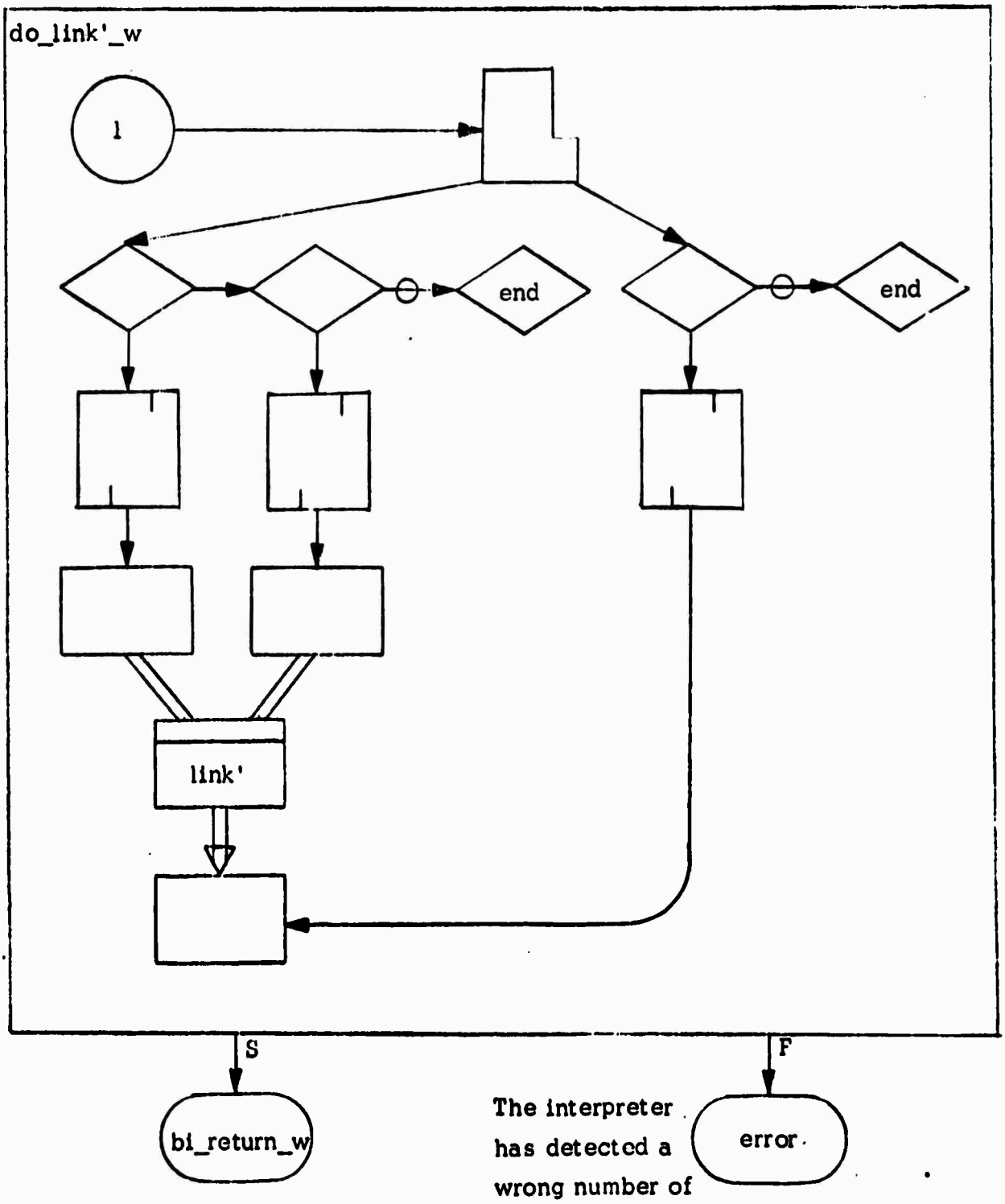
The interpreter has detected a wrong number of tails or heads on a write-call on the builtin "link".



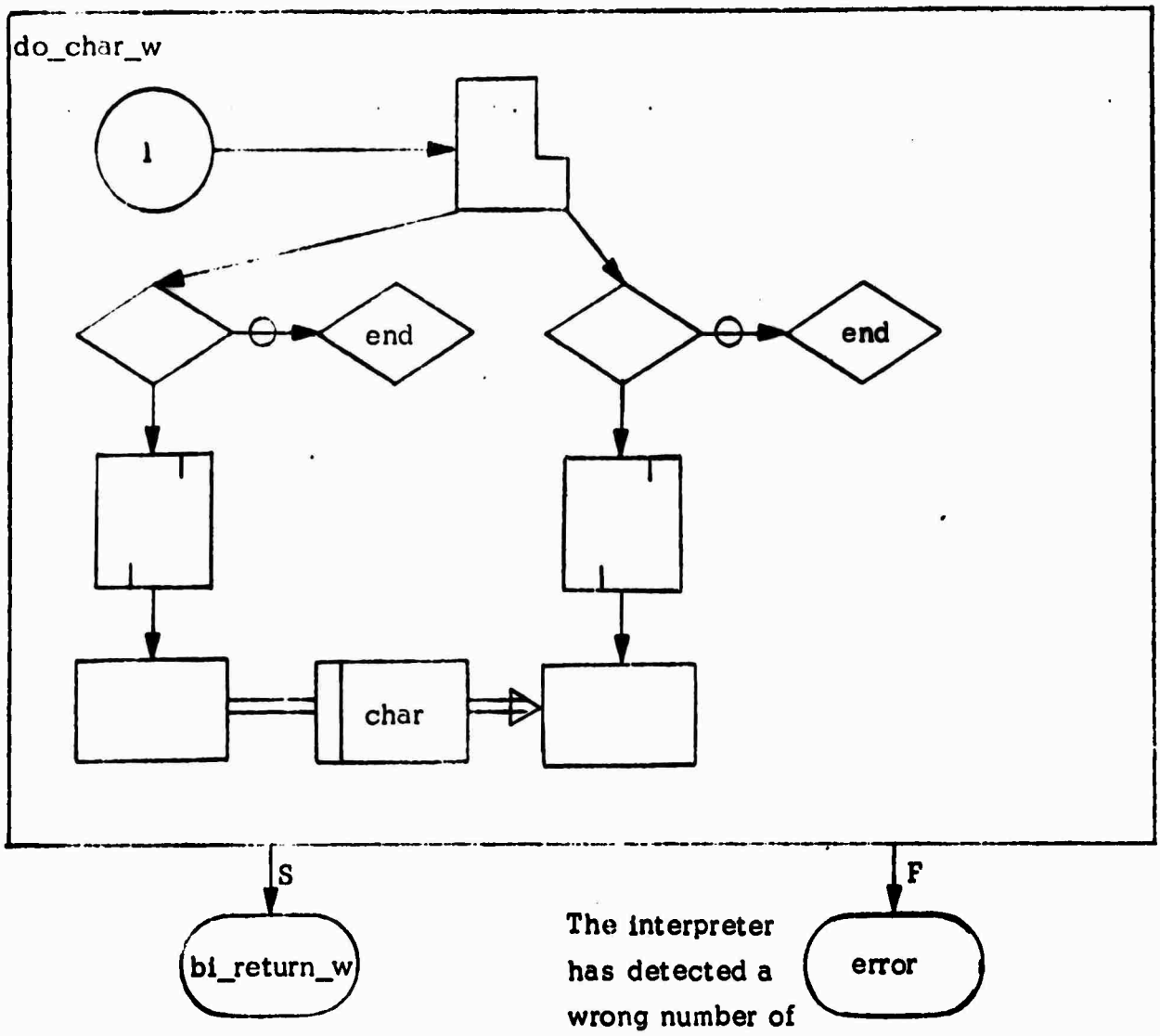
The interpreter has detected a wrong number of tails or heads on a write-call on the builtin "read_function".



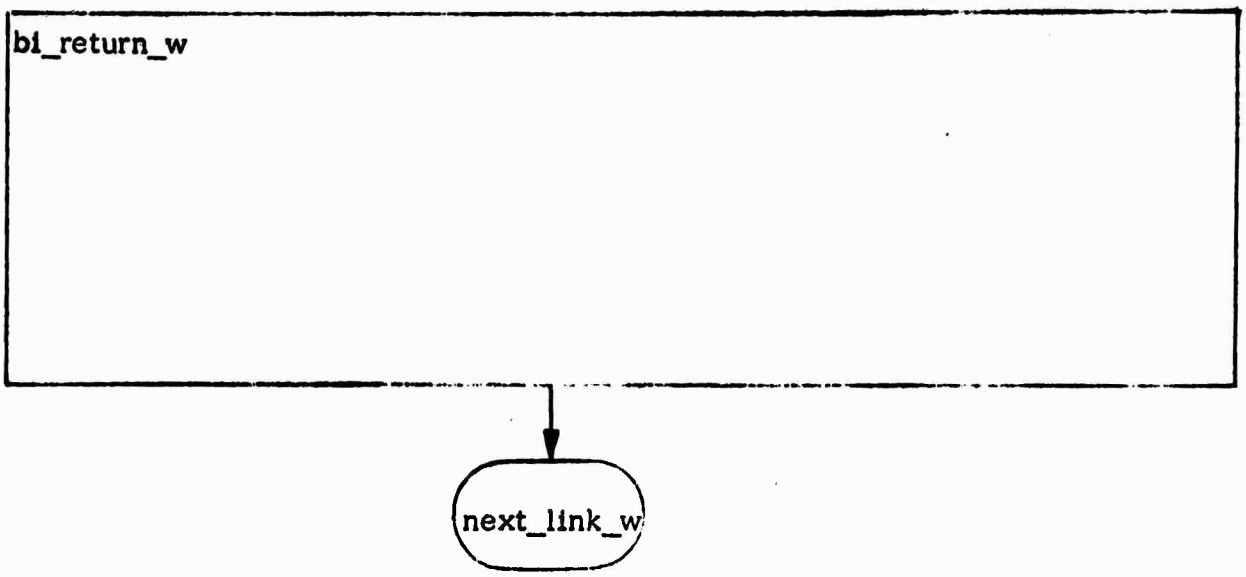
The interpreter has detected a wrong number of tails or heads on a write-call on the builtin "write_function".

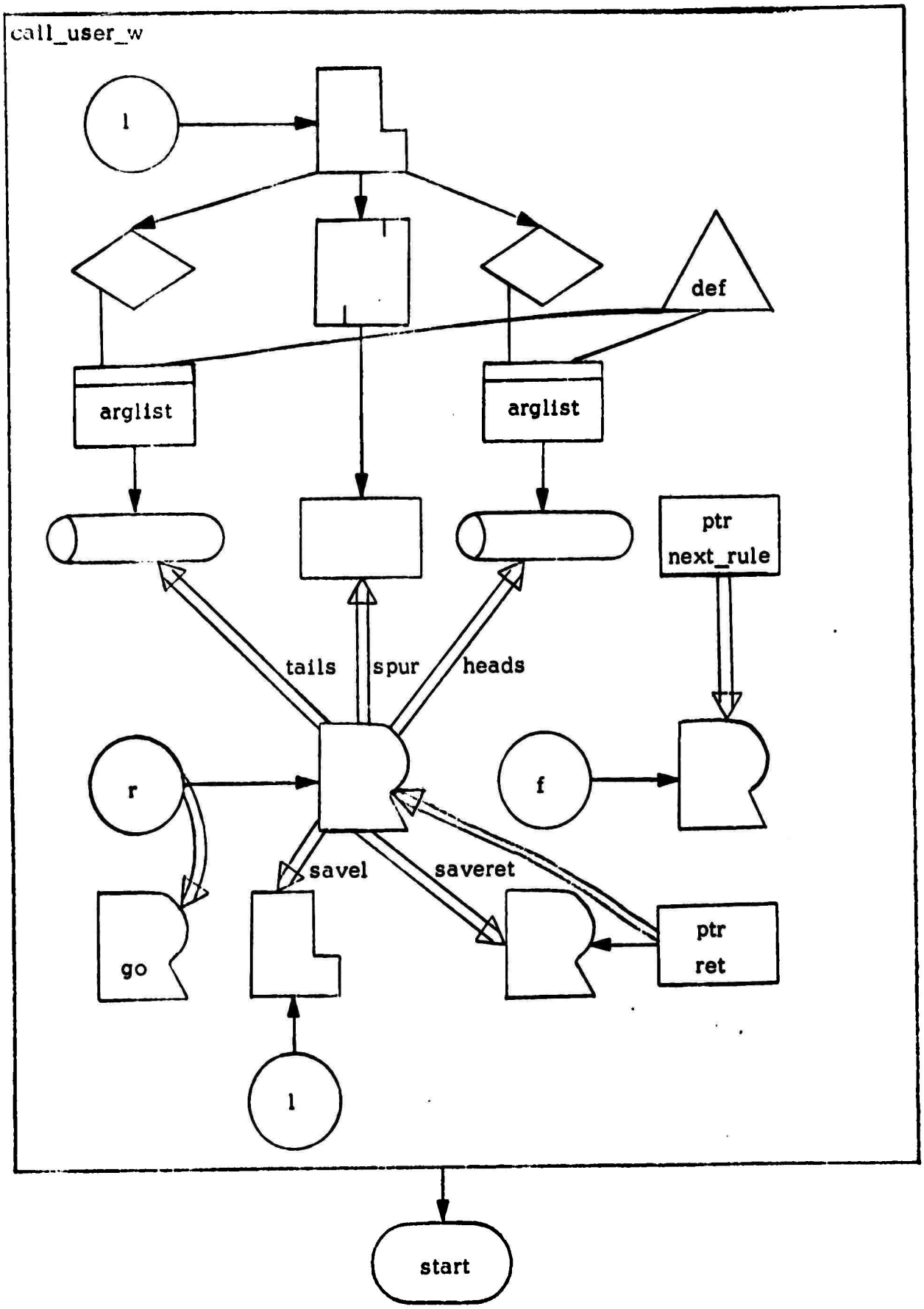


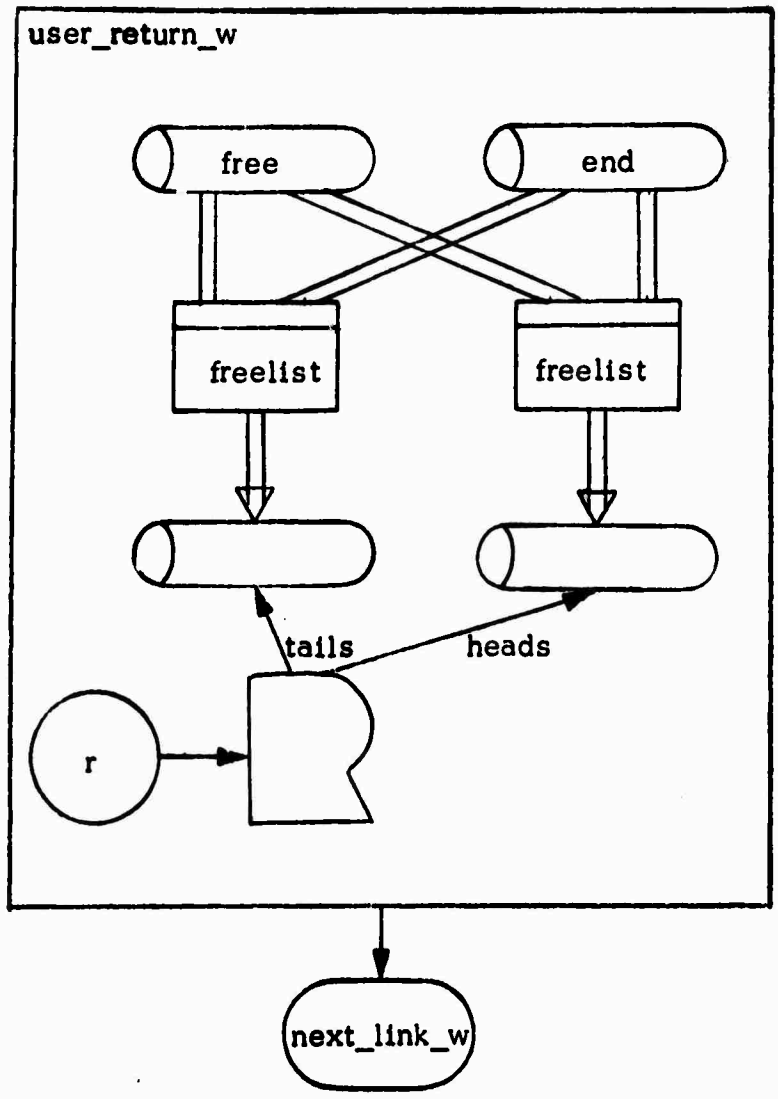
The interpreter has detected a wrong number of tails or heads on a write-call on the builtin "link'".

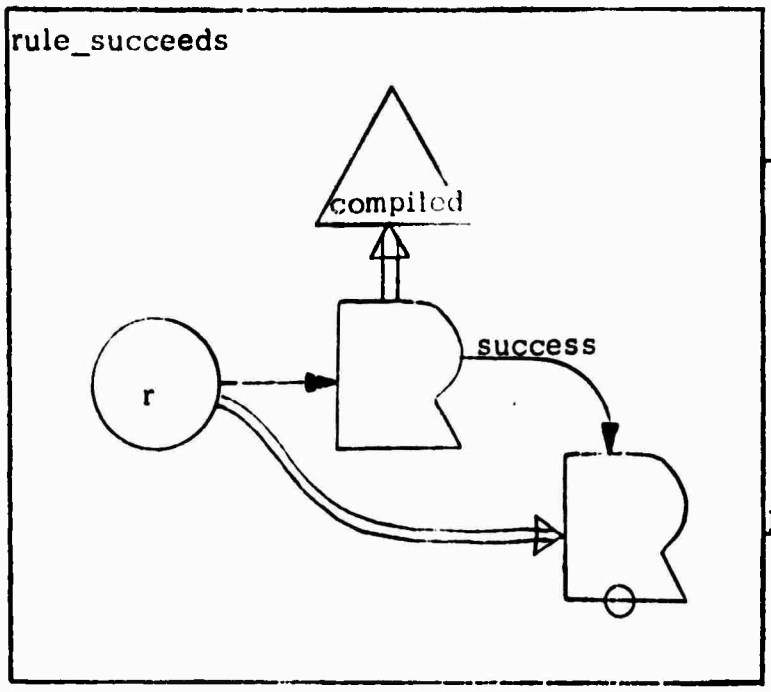


The interpreter has detected a wrong number of tails or heads on a write-call on the builtin "char".

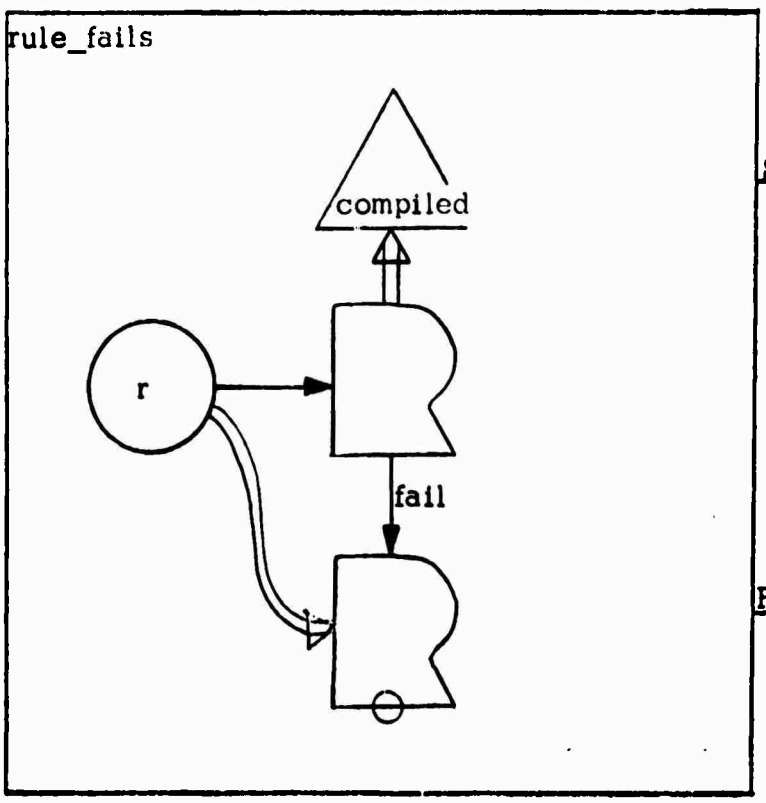

bi_return_w
next_link_w


call_user_w
l
def
arglist
arglist
ptr
next_rule
tails
spur
heads
r
f
savel
saveret
go
ptr
ret
l
start

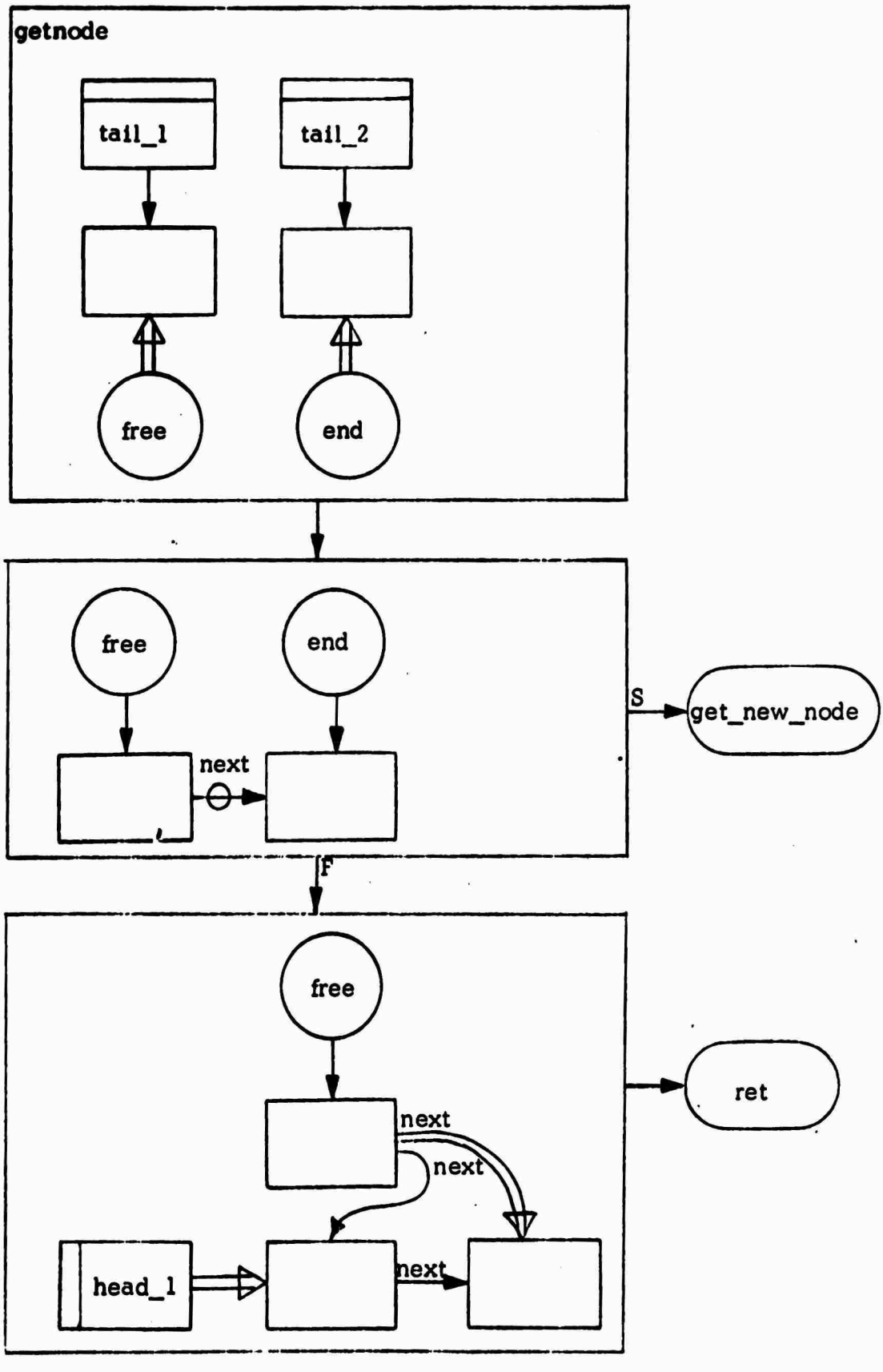

user_return_w
free
end
freelist
freelist
tails
heads
r
next_link_w

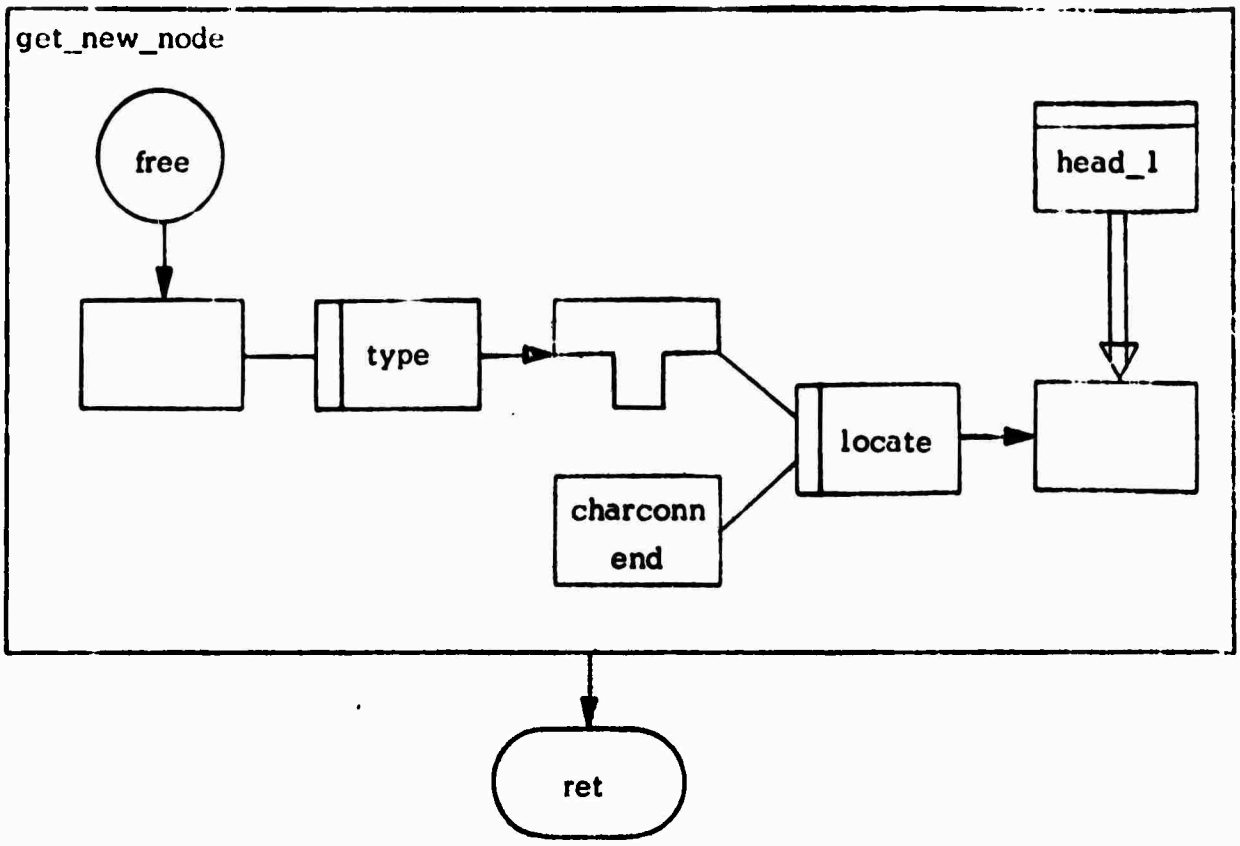


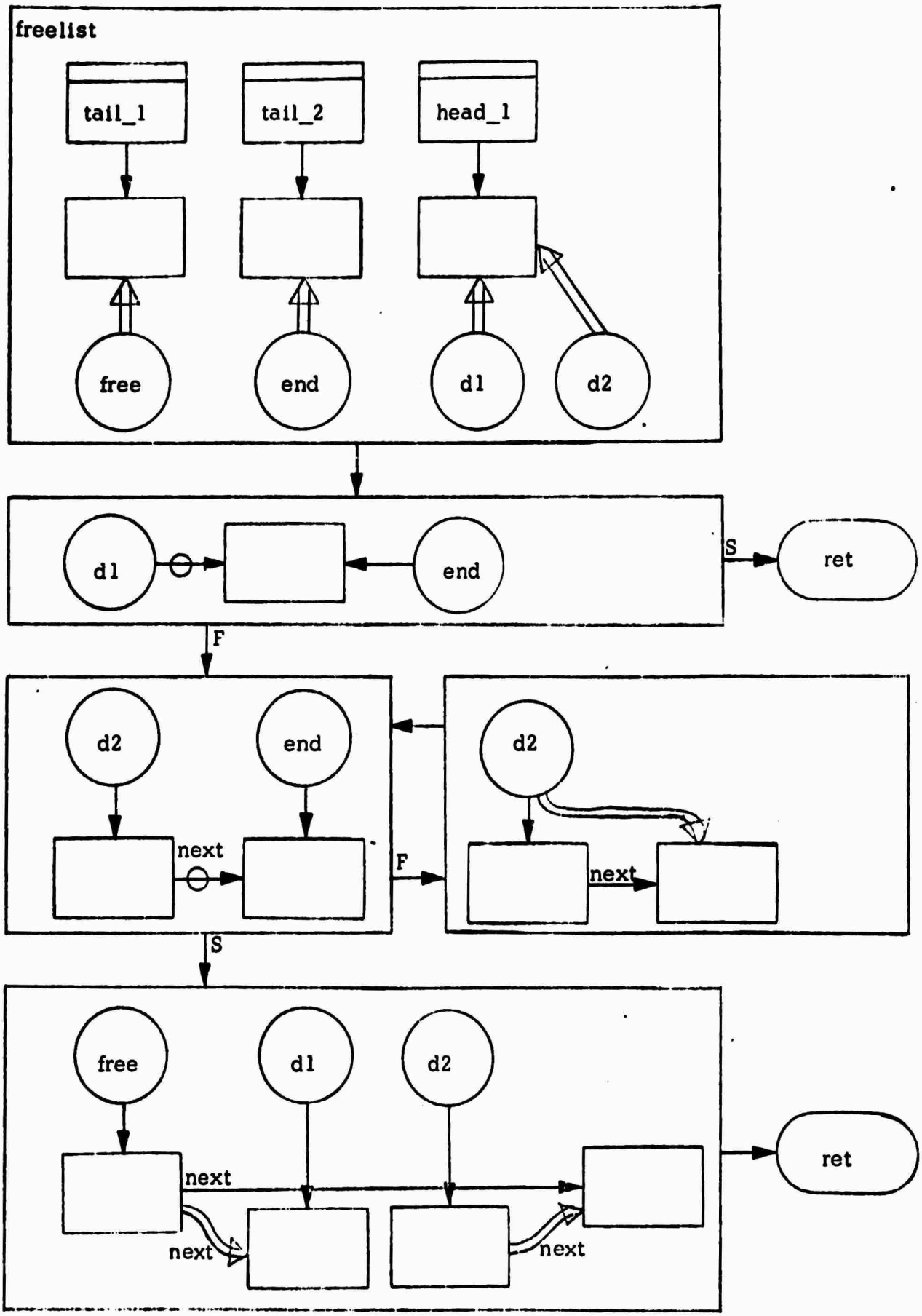
The interpreter has detected the "success" exit from a rule leads to " ... " , which is not a rule.

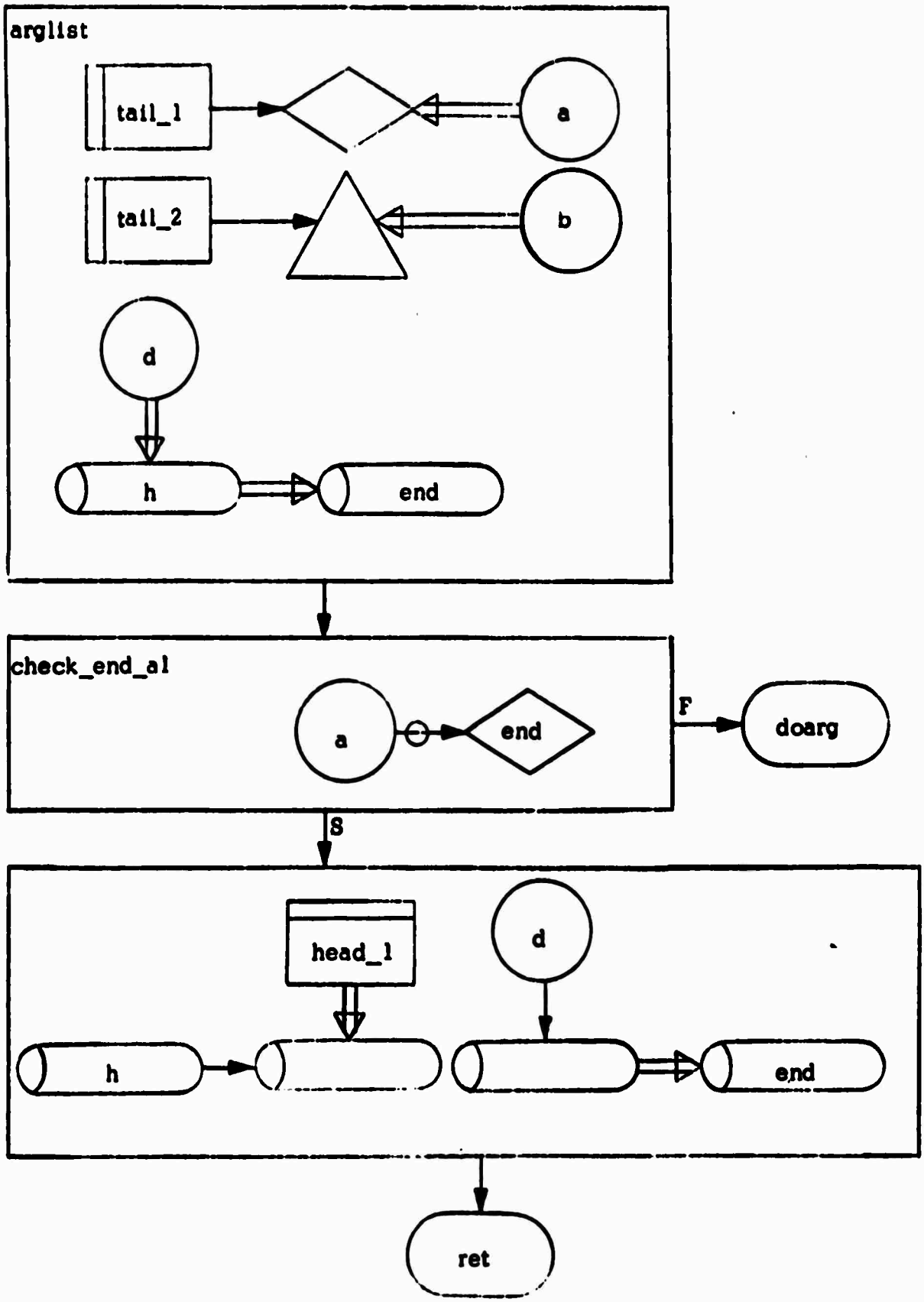


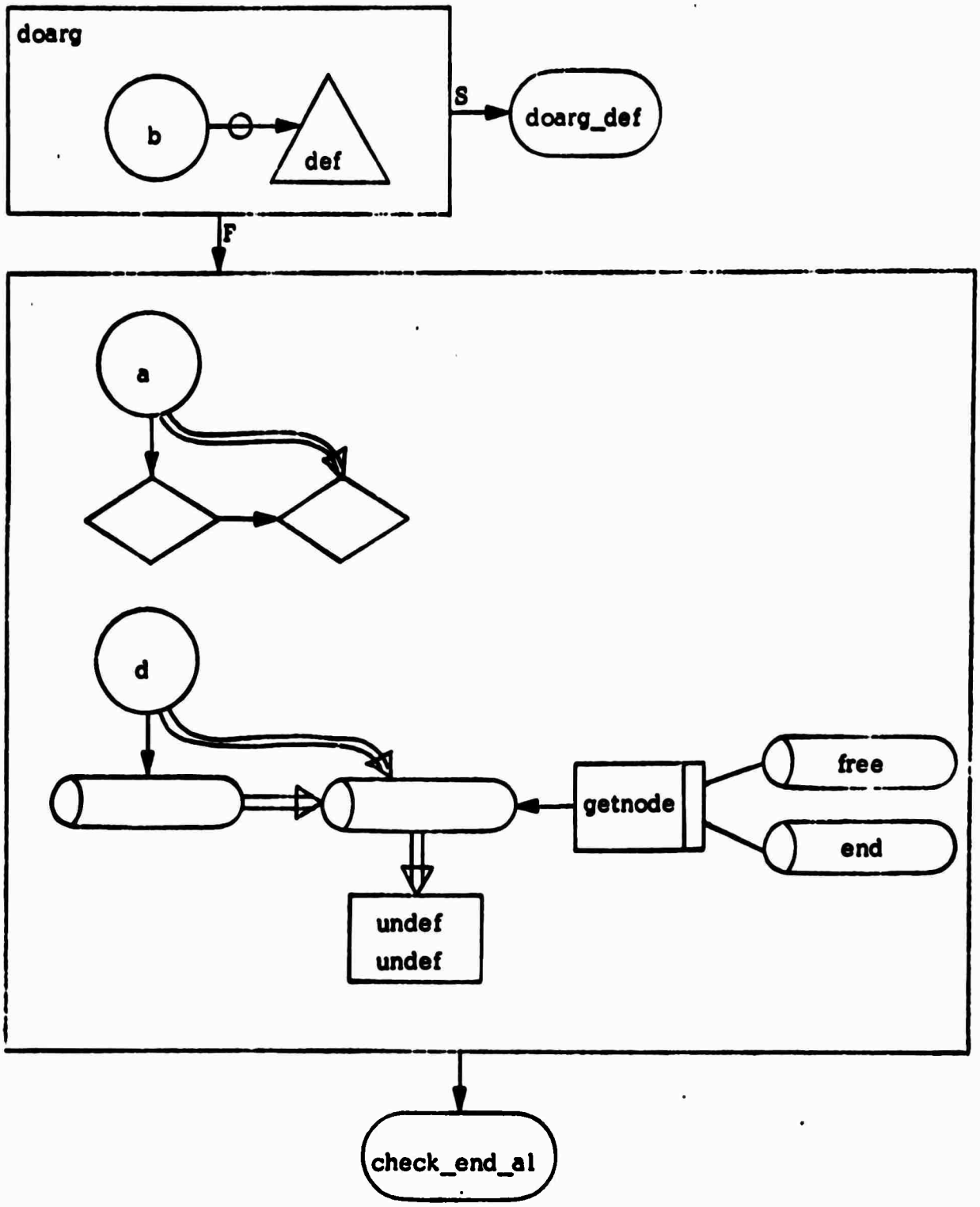
The interpreter has detected the "fail" exit from a rule leads to " ... " , which is not a rule.

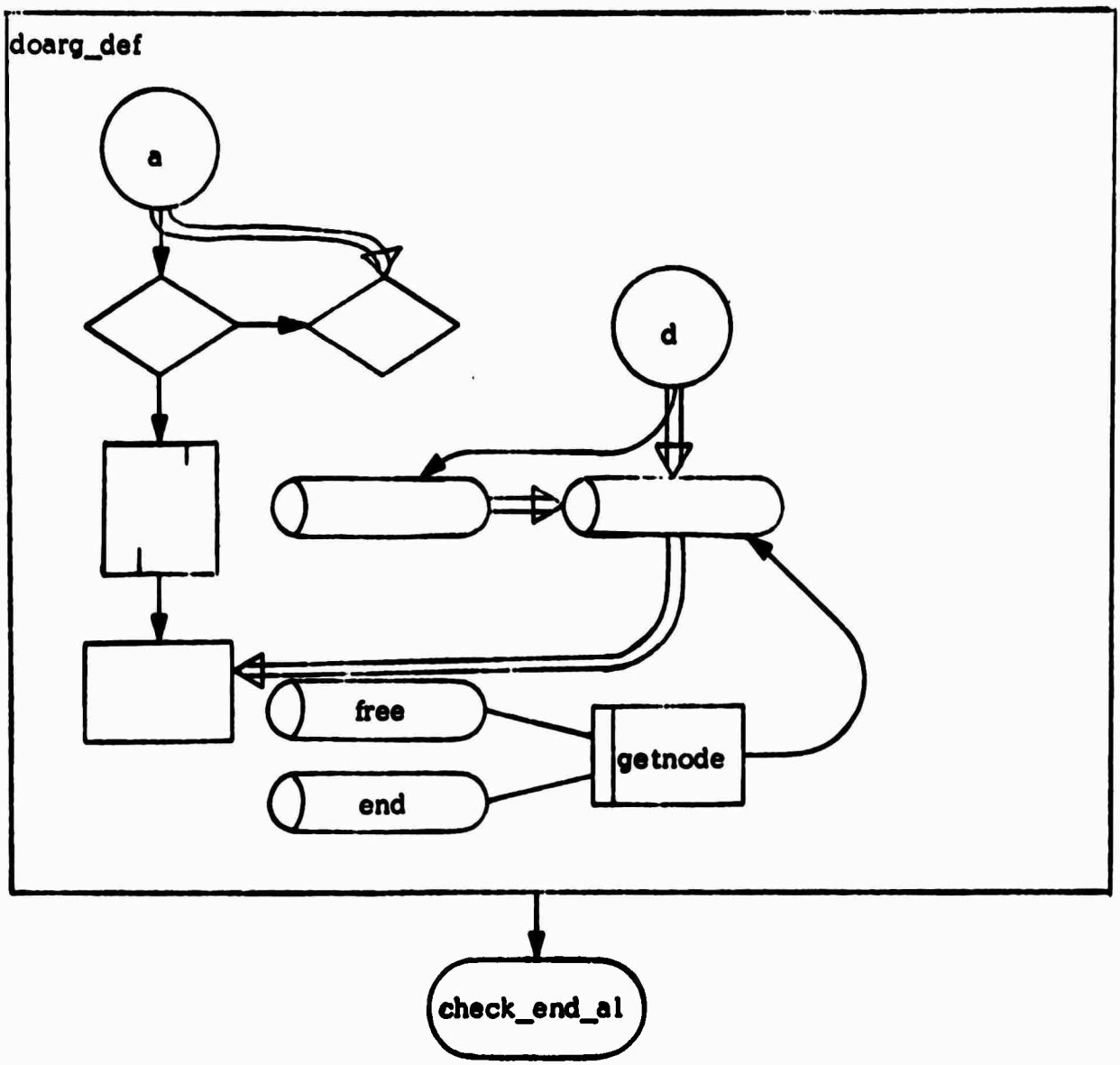

getnode
tail_1
tail_2
free
end
free
end
next
S
get_new_node
F
free
next
next
ret
head_1
next

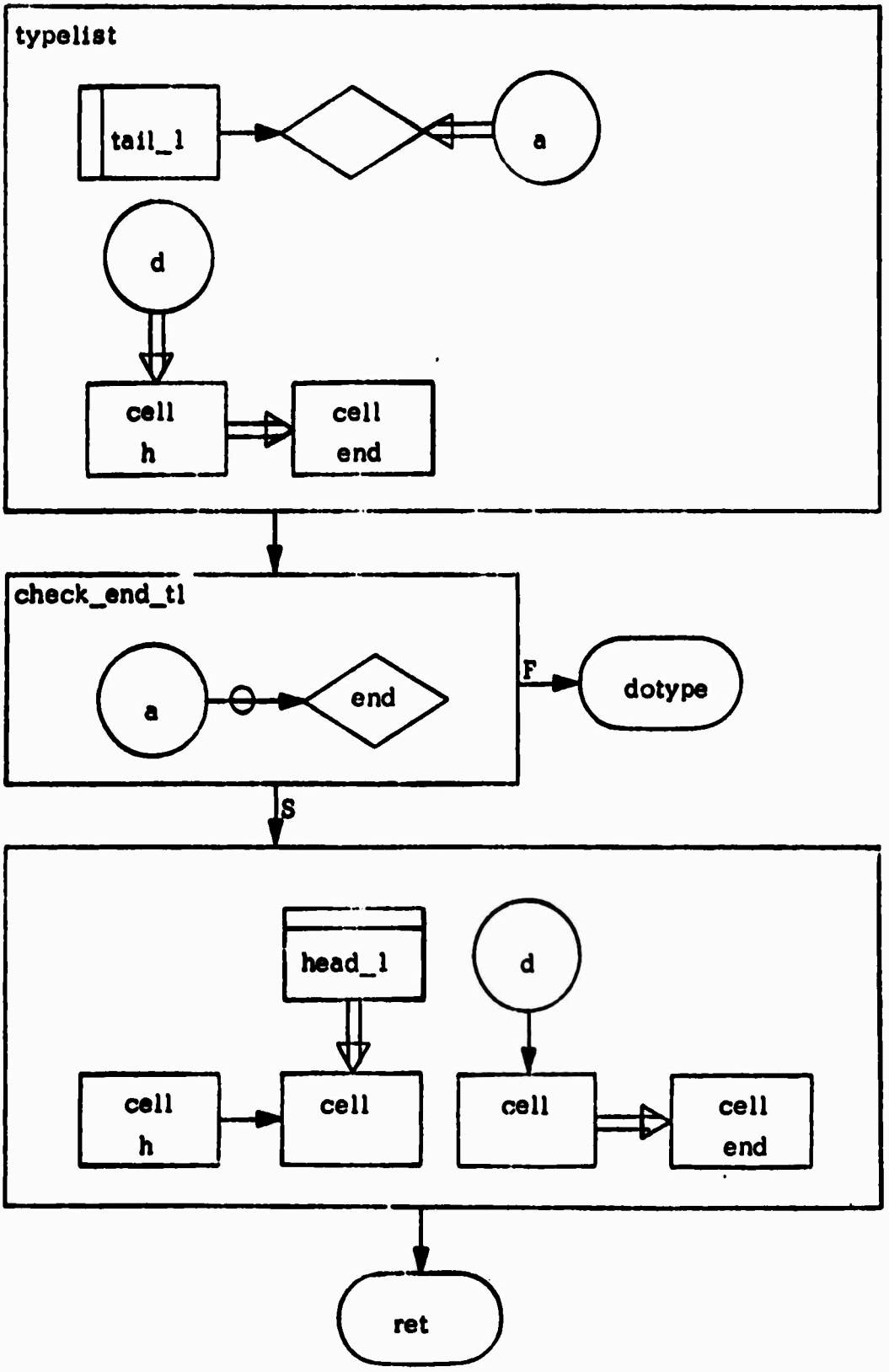

get_new_node
free
head_1
type
locate
charconn
end
ret

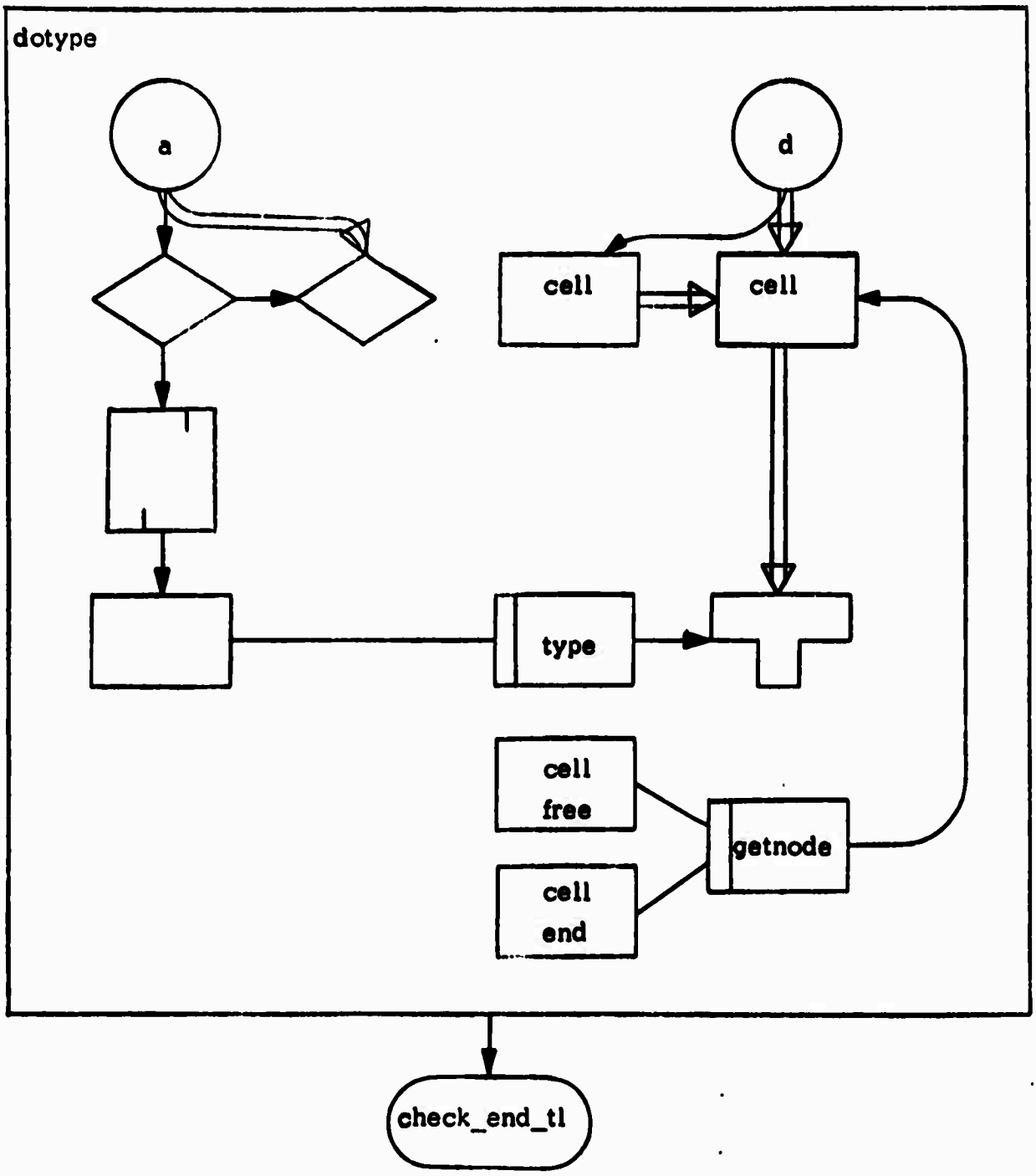

freelist
tail_1
tail_2
head_1
free
end
d1
d2
d1
end
S
ret
F
d2
end
next
F
d2
next
S
free
d1
d2
next
next
next
ret

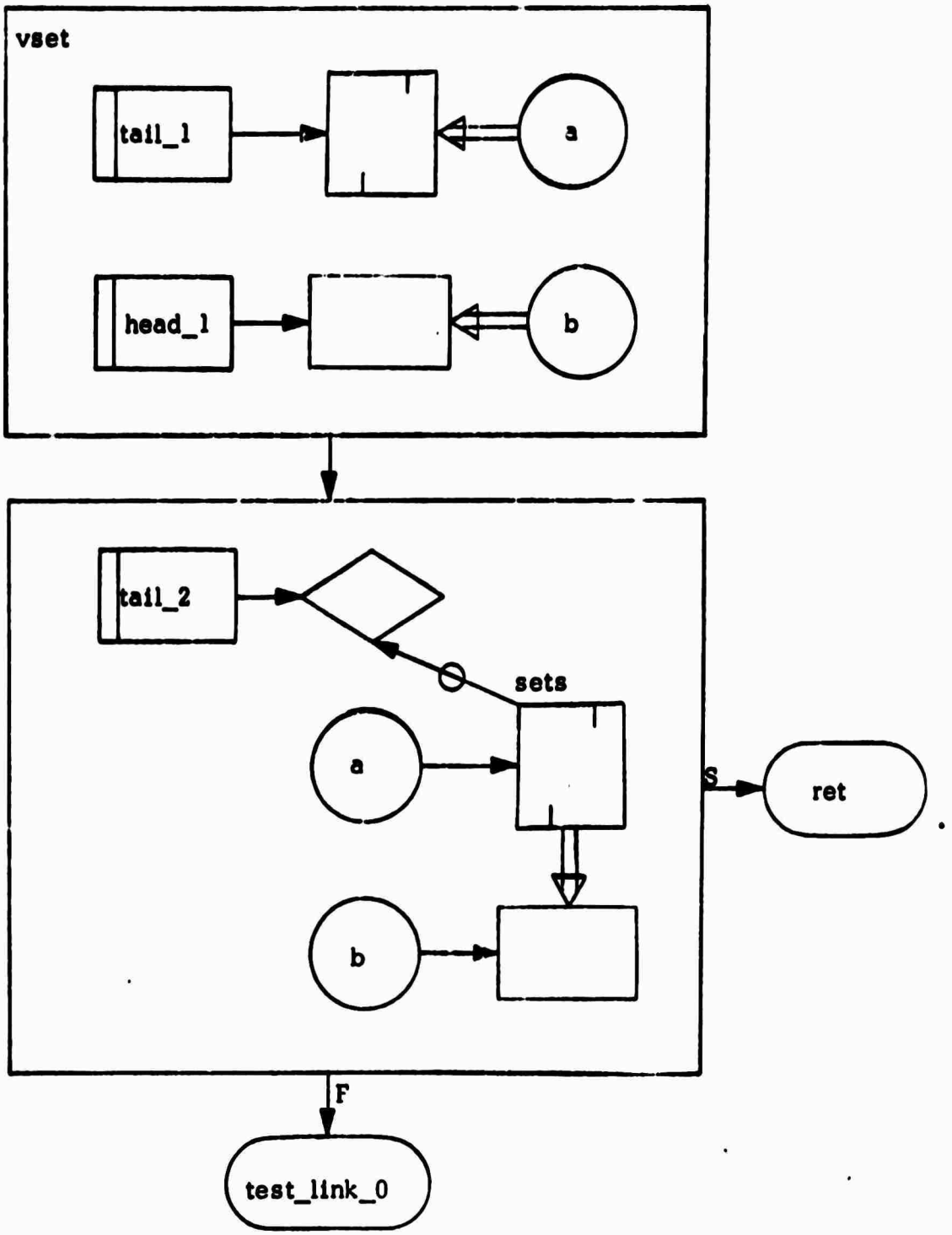

arglist
tail_1
a
tail_2
b
d
h
end
check_end_al
a
end
F
doarg
S
head_1
d
h
end
ret

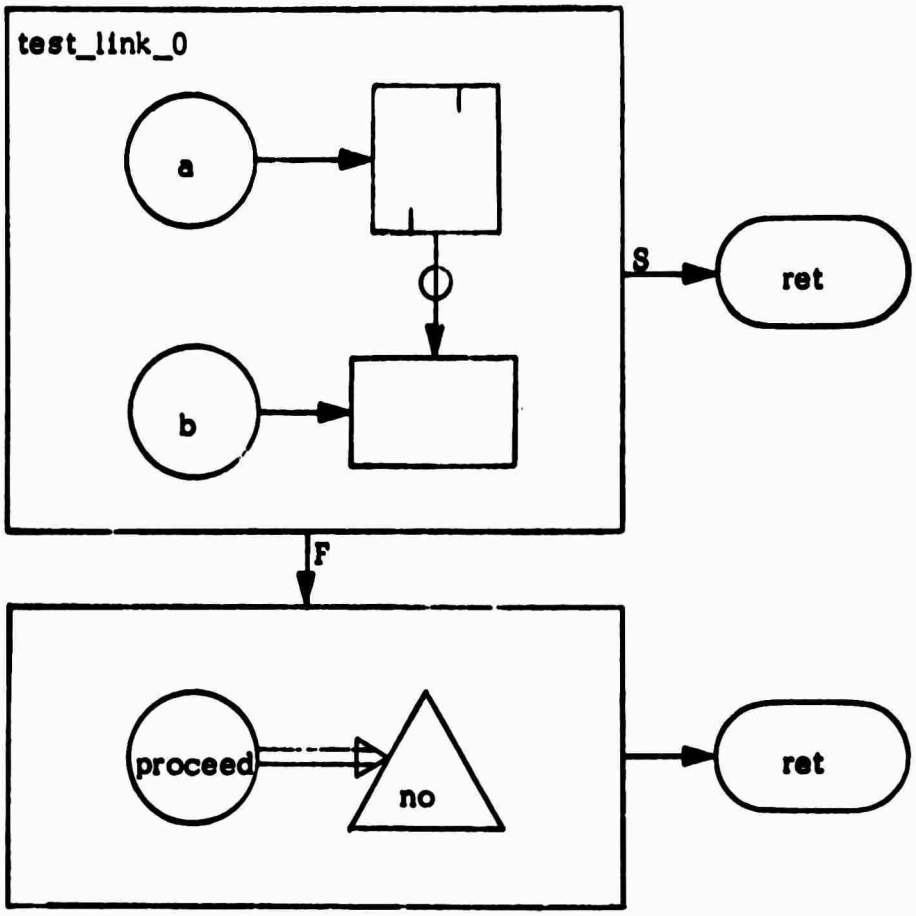

doarg
b
def
S
doarg_def
F
a
d
getnode
free
end
undef
undef
check_end_al

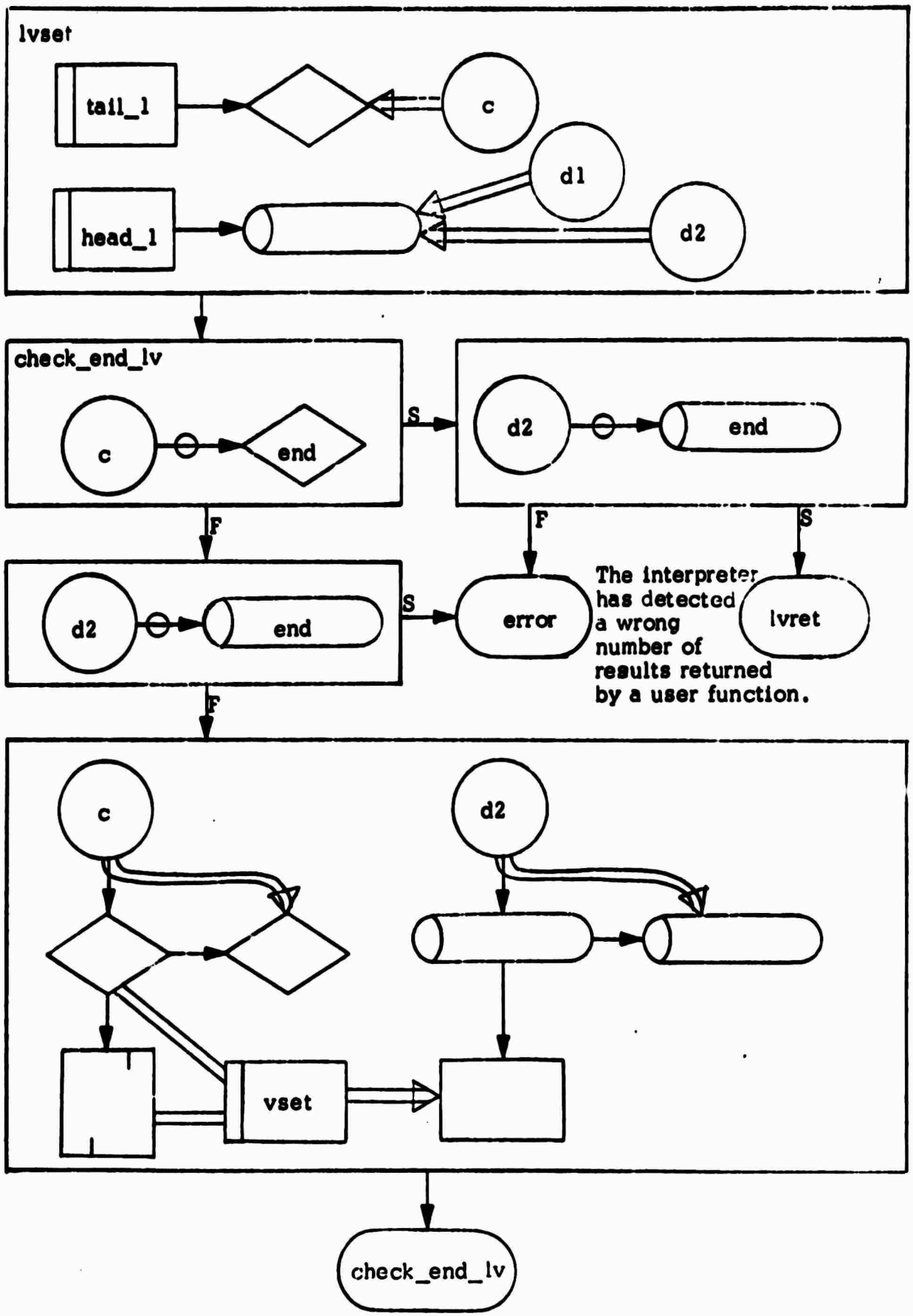

doarg_def
a
d
free
getnode
end
check_end_al

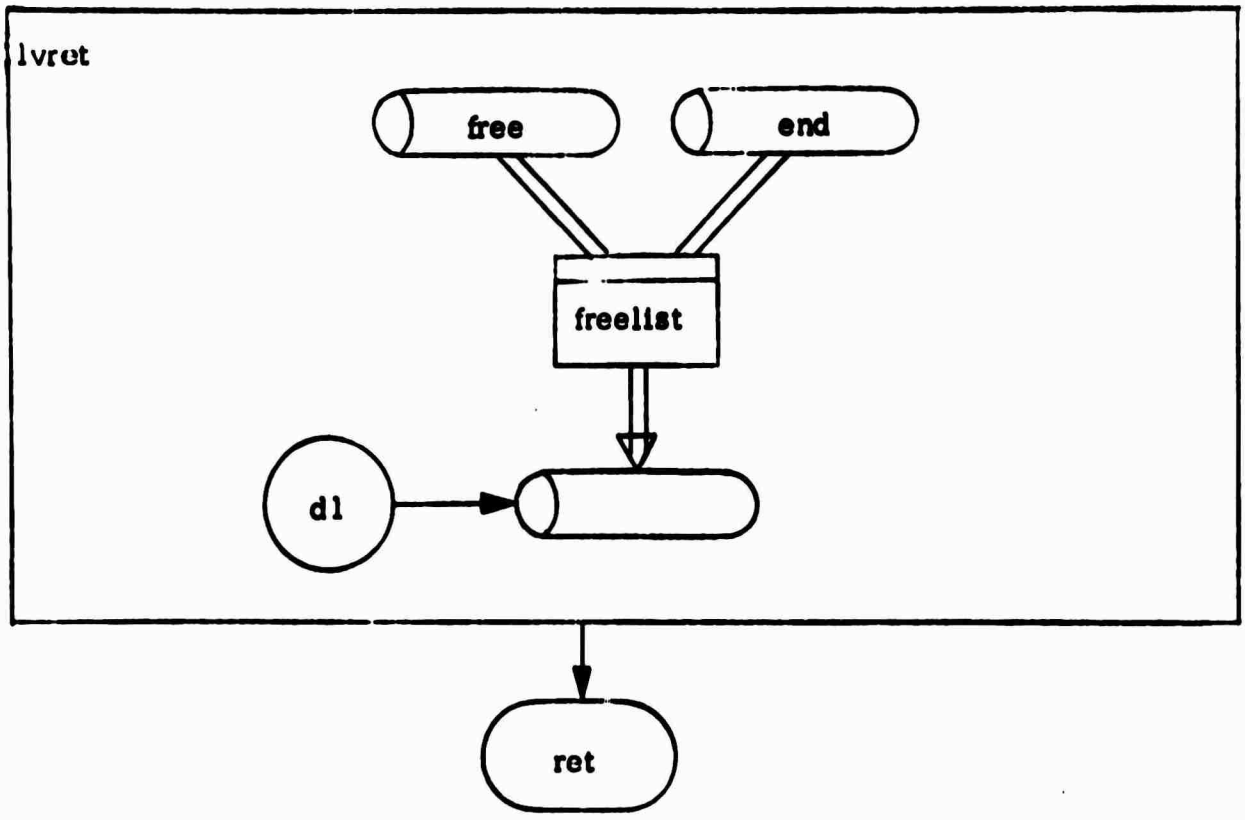

typelist
tail_l
a
d
cell
h
cell
end
check_end_tl
a
end
F
dotype
S
head_l
d
cell
h
cell
cell
cell
end
ret

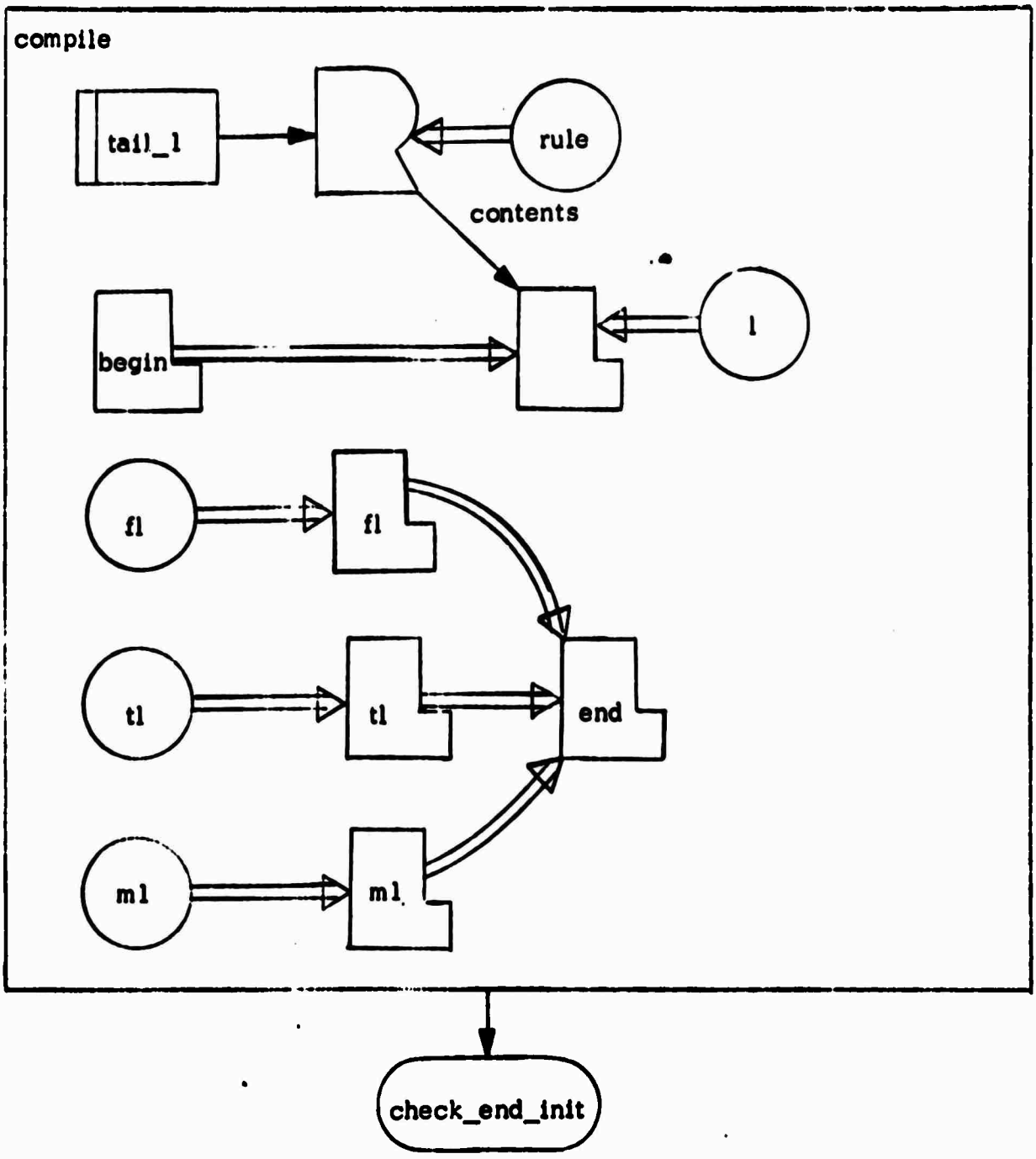

dotype
a
d
cell
cell
type
cell
free
cell
end
getnode
check_end_tl

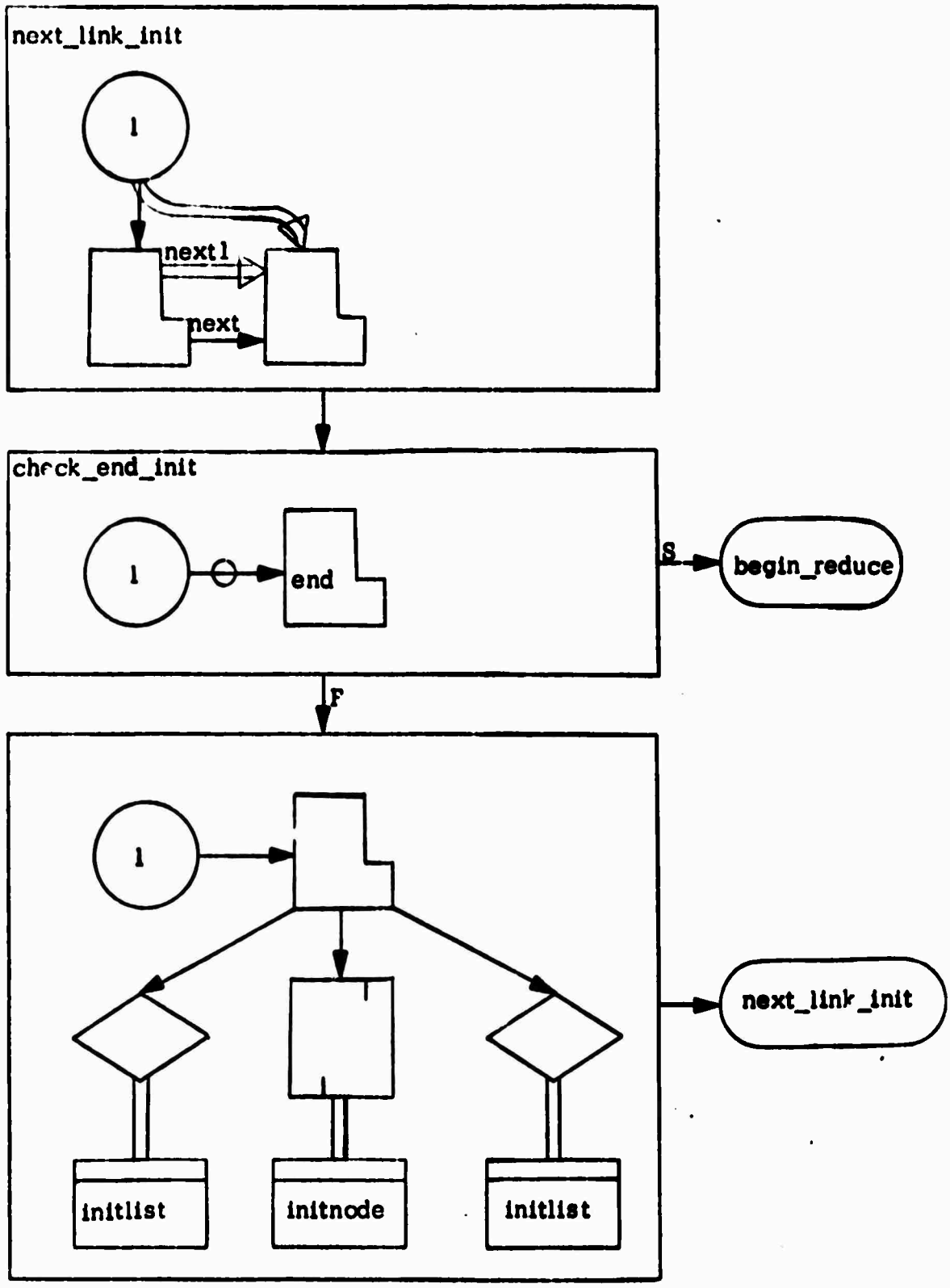

vset
tail_1
a
head_1
b
tail_2
sets
a
S
ret
b
F
test_link_0

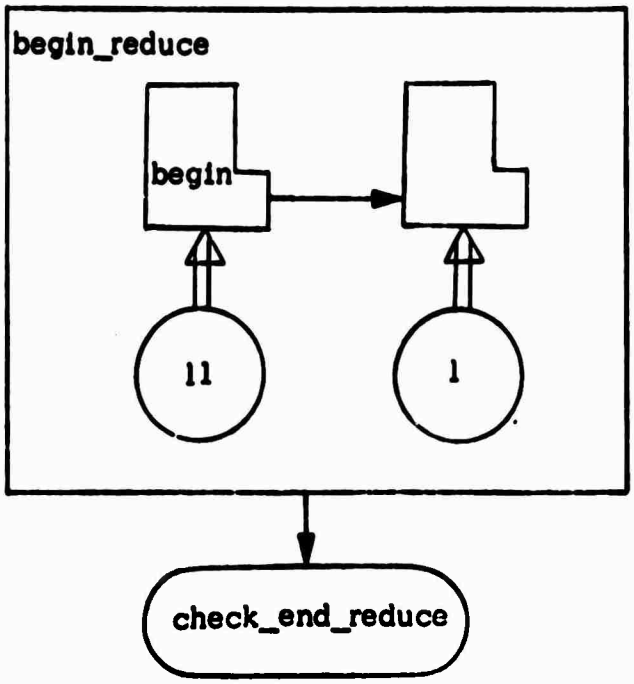

test_link_0
a
b
S
ret
F
proceed
no
ret

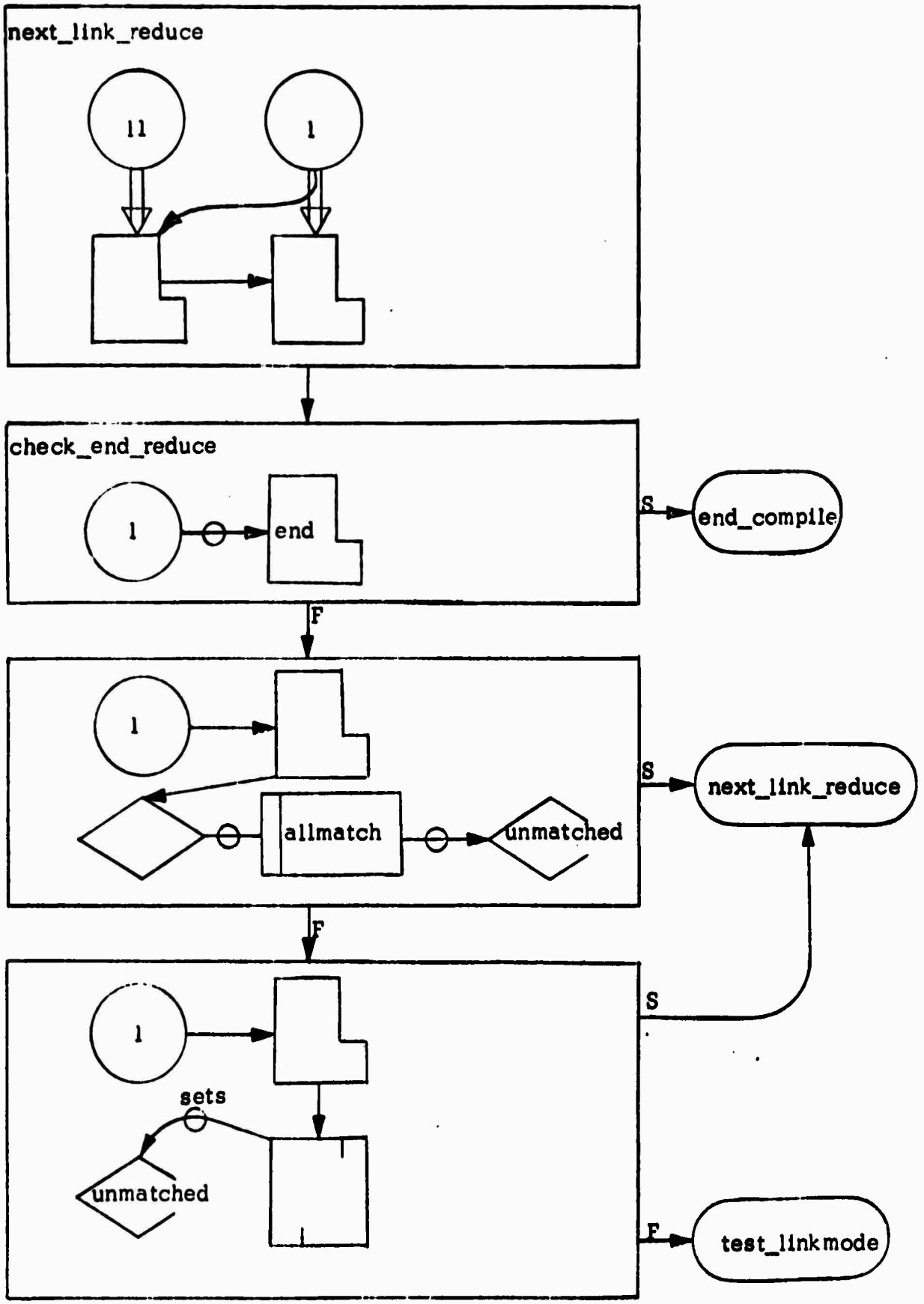

lvset
tail_1
c
d1
head_1
d2
check_end_lv
c
end
S
d2
end
F
F
S
d2
end
S
error
The interpreter has detected a wrong number of results returned by a user function.
lvret
F
c
d2
vset
check_end_lv

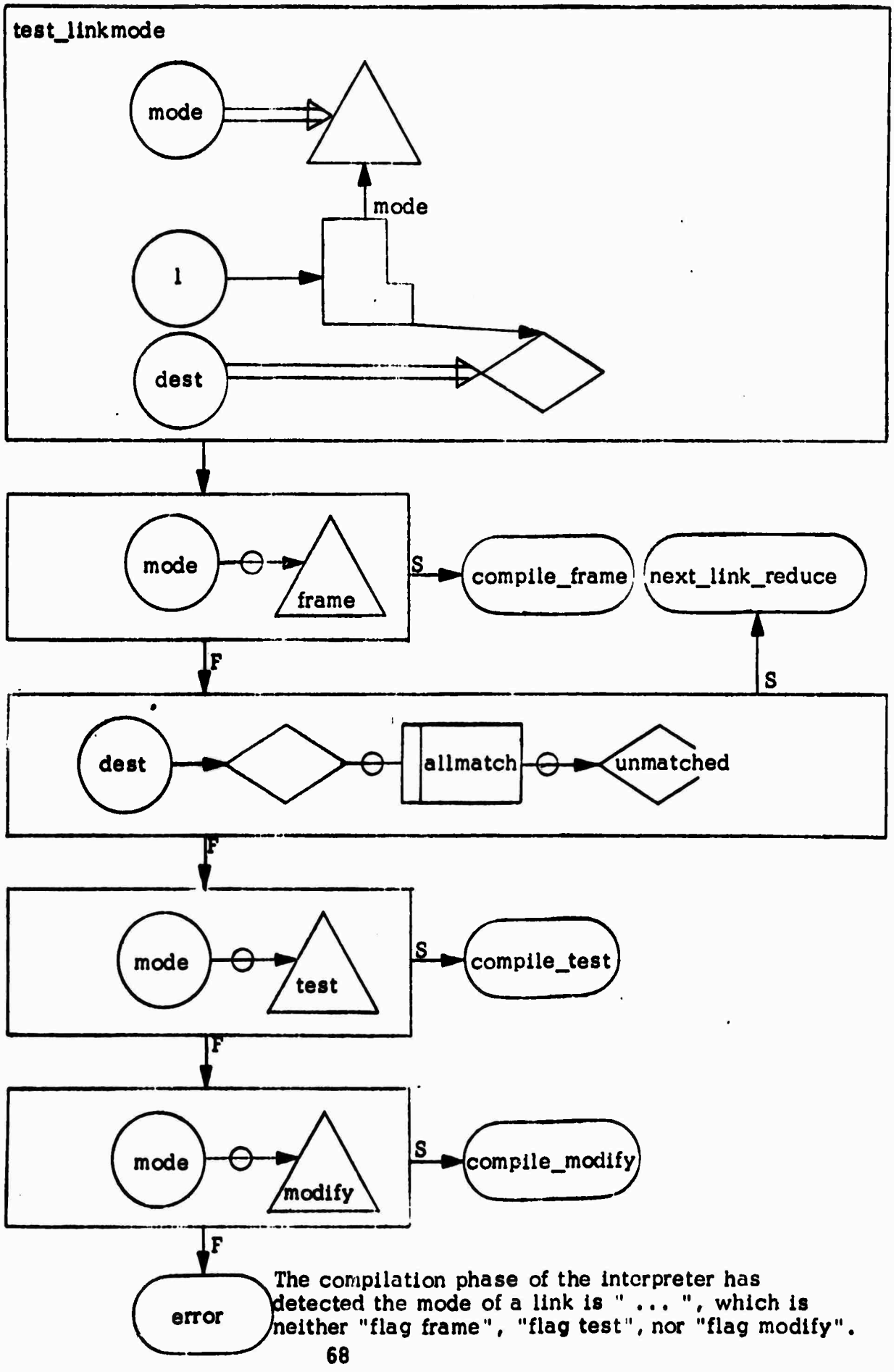

lvret
free
end
freelist
dl
ret


compile
tail_1
rule
contents
begin
l
f1
f1
t1
t1
end
m1
m1
check_end_init

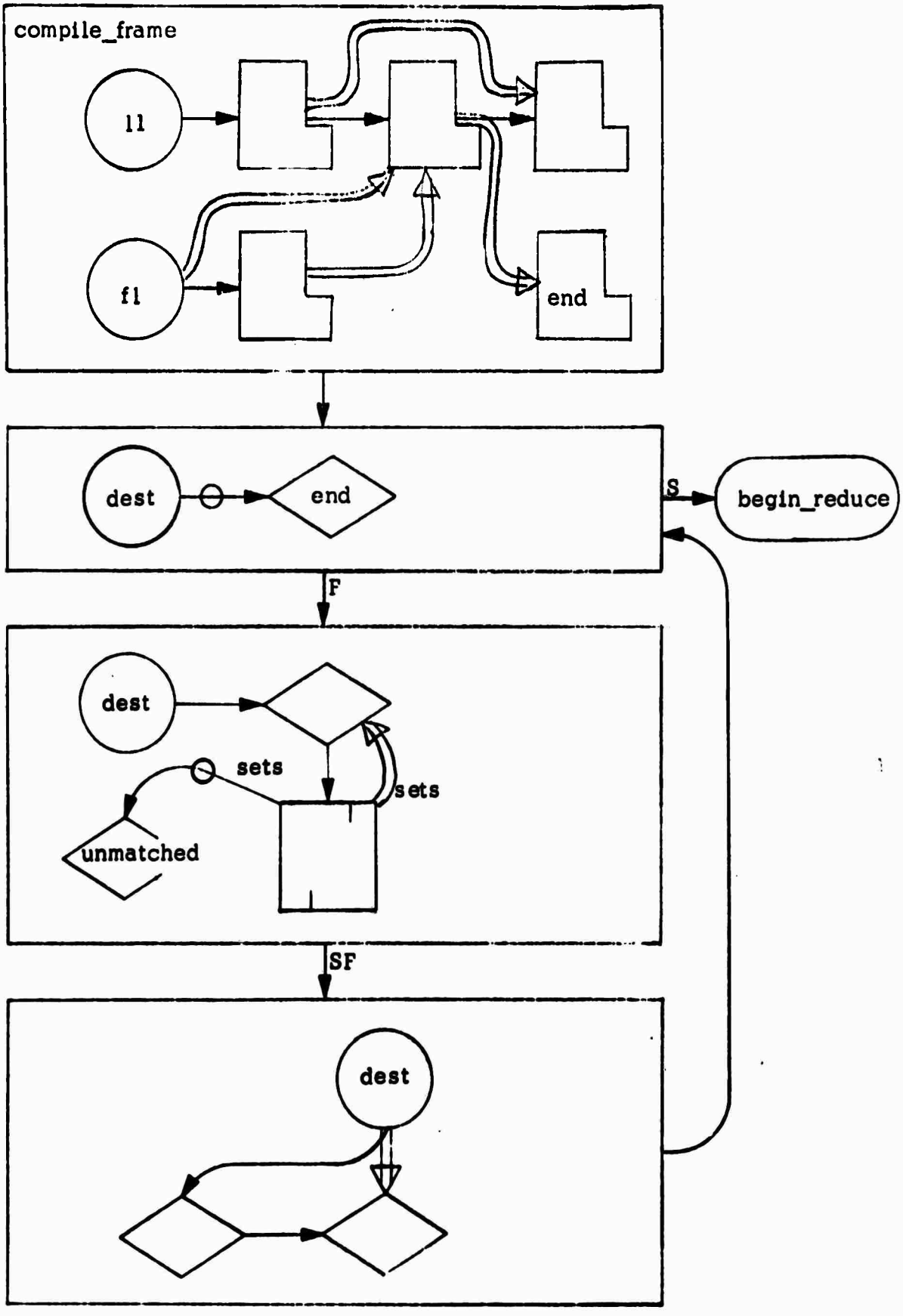

next_link_init
1
next1
next
check_end_init
1
end
S
begin_reduce
F
1
initlist
initnode
initlist
next_link_init

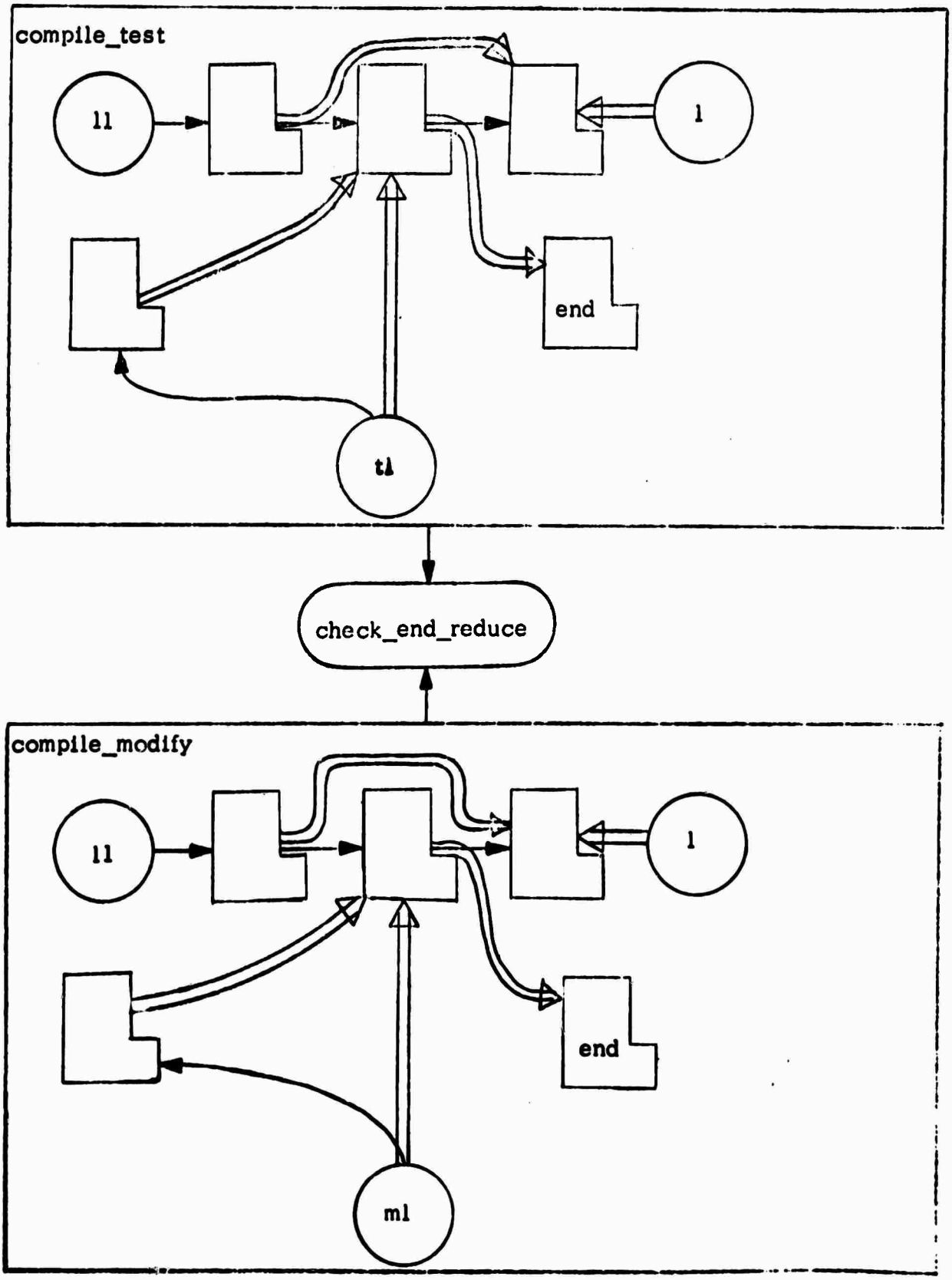

begin_reduce
begin
11
1
check_end_reduce

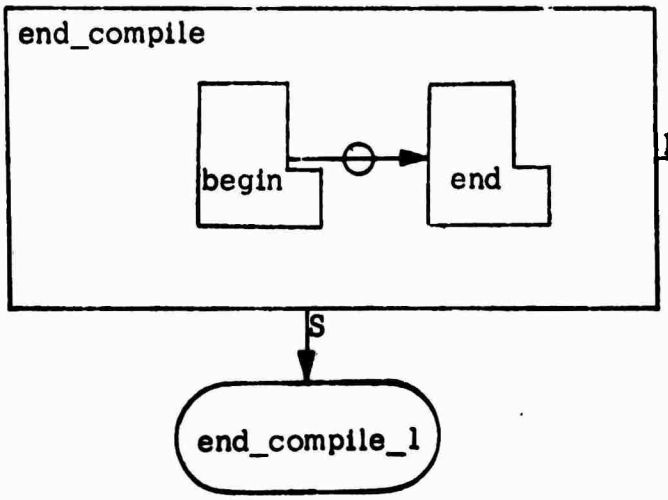

next_link_reduce
11
1
check_end_reduce
1
end
S
end_compile
F
1
allmatch
unmatched
S
next_link_reduce
F
1
sets
unmatched
S
F
test_linkmode



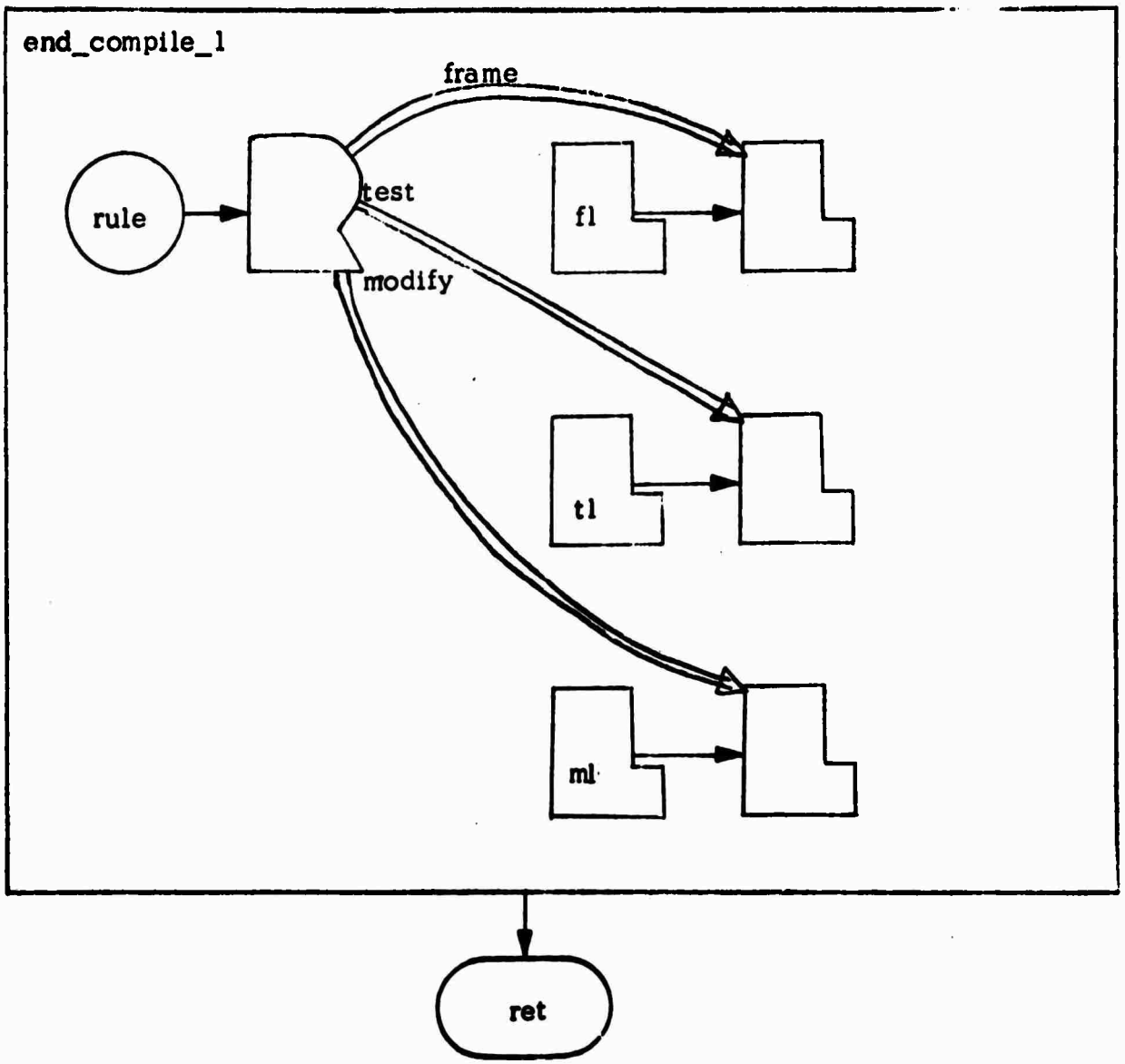
The compilation phase of the interpreter has detected the mode of a link is " ... ", which is neither "flag frame", "flag test", nor "flag modify".

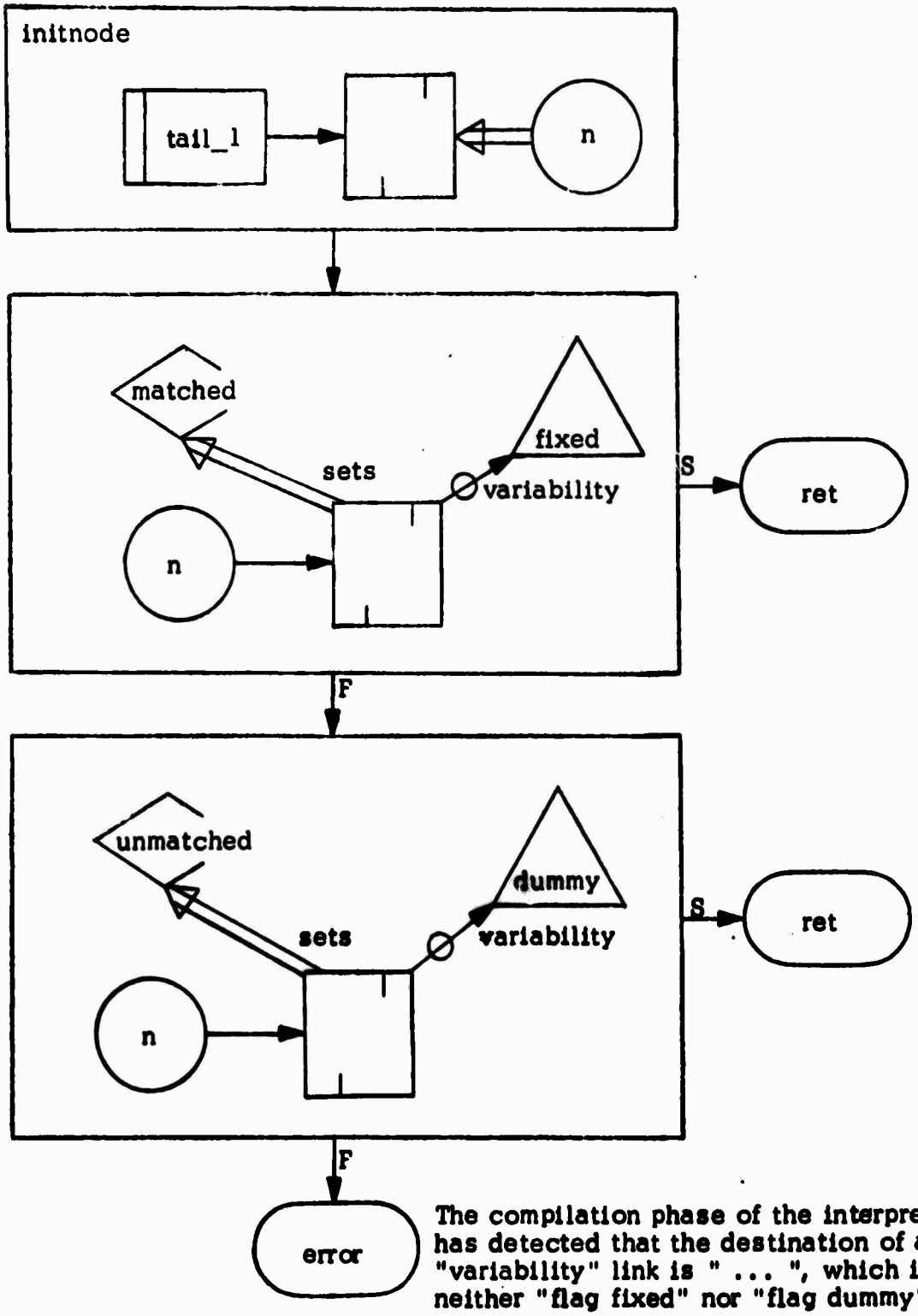

compile_frame
11
f1
end
dest
end
S
begin_reduce
F
dest
sets
sets
unmatched
SF
dest


compile_test
l1
1
end
t1
check_end_reduce
compile_modify
l1
1
end
m1

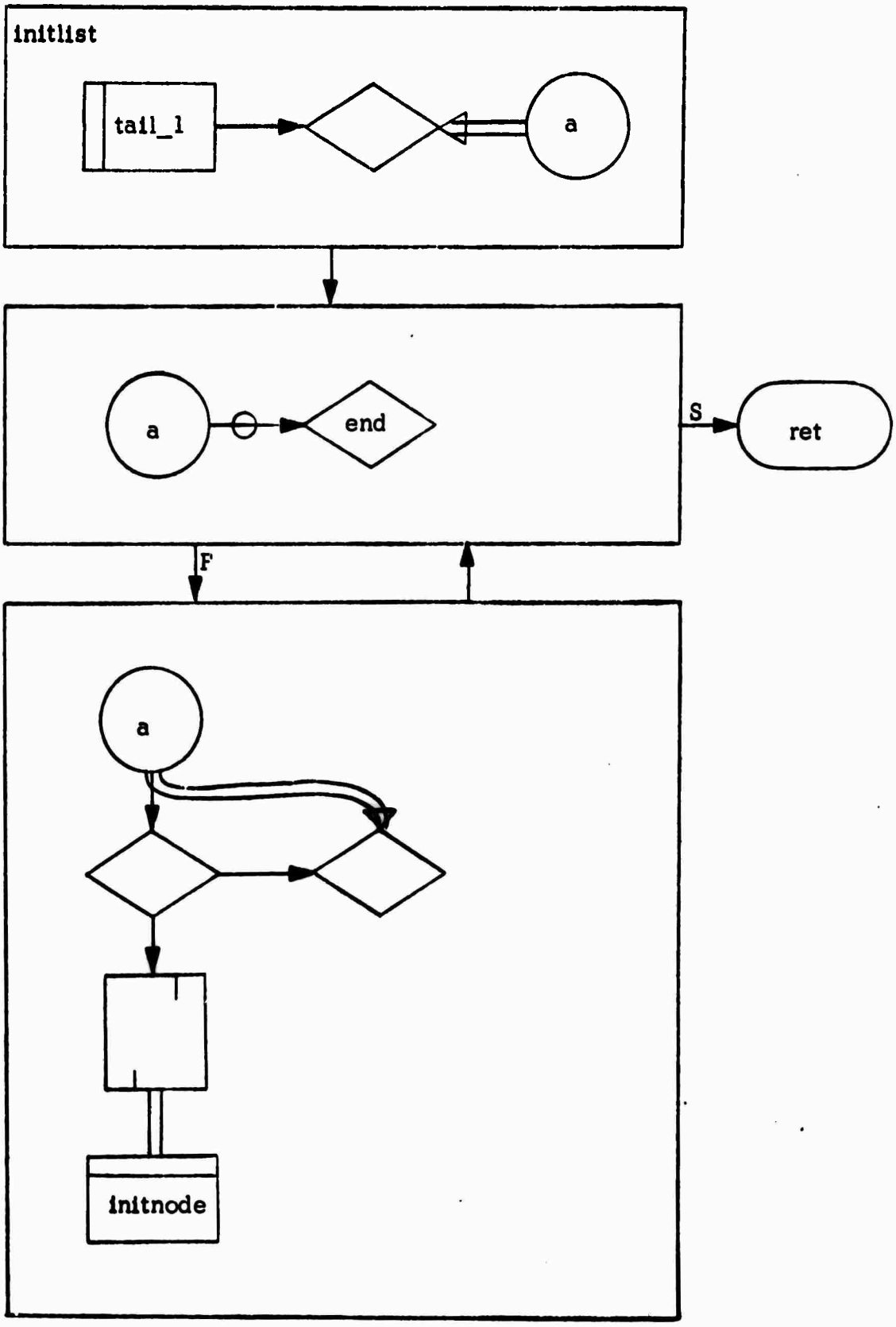


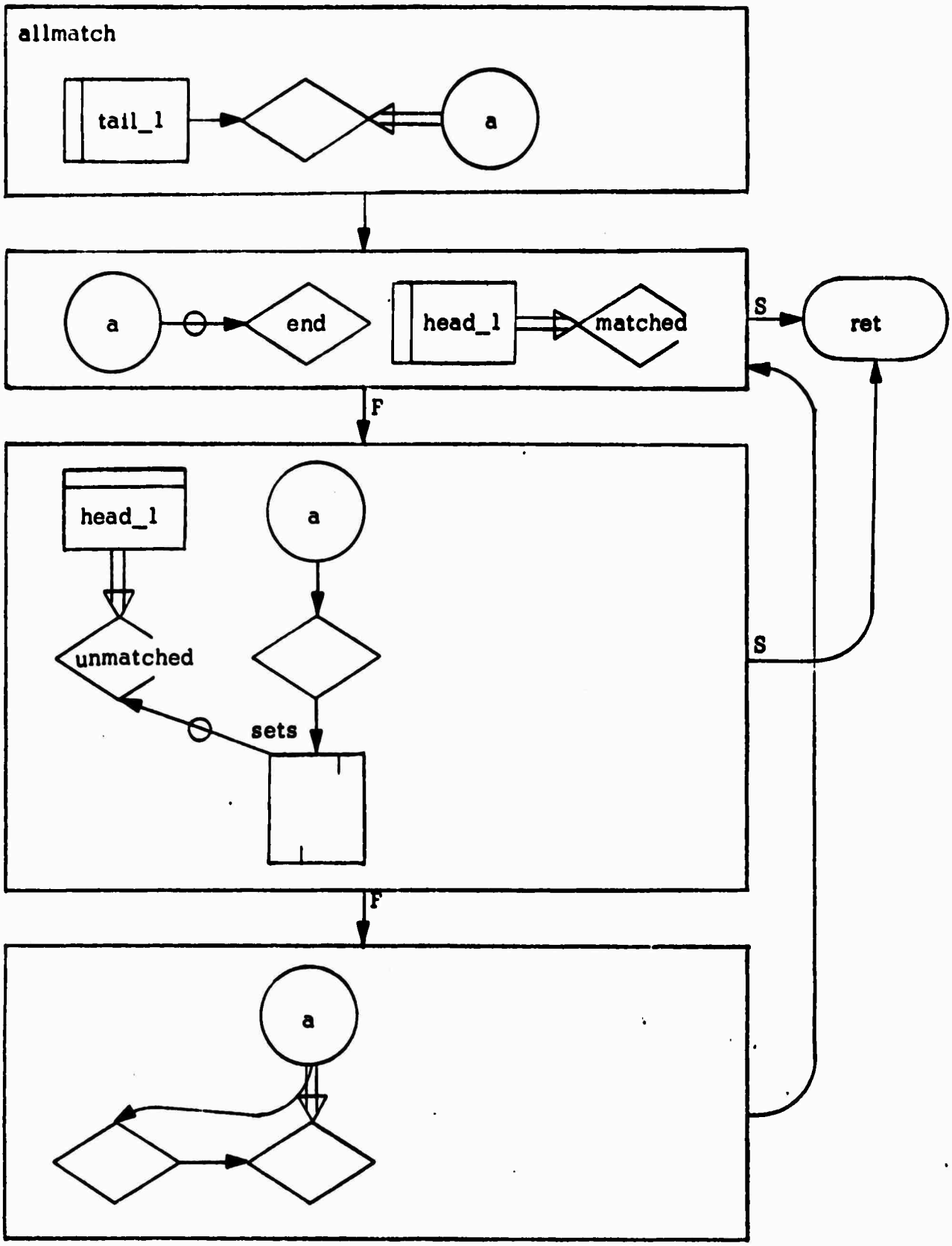
The compilation phase of the interpreter has detected that a rule contains an unreachable node.


end_compile_1
frame
rule
test
modify
fl
tl
ml
ret



The compilation phase of the interpreter has detected that the destination of a "variability" link is " ... ", which is neither "flag fixed" nor "flag dummy".


initlist
tail_1
a
a
end
S
ret
F
a
initnode


allmatch
tail_1
a
a
end
head_1
matched
S
ret
F
head_1
a
unmatched
sets
S
F
a